Technology Enhanced Learning Living lab for Manufacturing Environments

D2.1

TELL ME RW/DW/VW Platforms – Issue 1

M12 issue

Document Owner:	Michele Sesana (TXT)
Contributors:	Stefano Bianchi (SOFT), Maurizio Megliola (PIKS), Jaakko Karjalainen (VTT)
Dissemination:	Public
Contributing to:	WP2
Date:	31/10/2013 (Due Date)
Revision:	0.9

VERSION HISTORY

	DATE	NOTES AND COMMENTS
0.1	01/10/2013	TOC PROPOSAL
0.2	04/10/2013	(TXT) DRAFT BASED ON WD2.1 CONTRIBUTIONS FROM ALL WP2 PARTNERS
0.3	16/10/2013	(TXT) UPDATED VERSION FOLLOWING GENOVA MEETING DISCUSSION
0.4	18/10/2013	(PIKS) UPDATE CONTRIBUTION TO VW
0.5	18/10/2013	(SOFT) REVIEW ON IOS AND ON ARCHITECTURE ANNEX
0.6	18/10/2013	(TXT) UPDATE CONTRIBUTION TO RW AND CONSOLIDATED VERSION
0.7	29/10/2013	(SOFT) TECHNICAL DETAILS ON IOS COMPONENTS
0.8	29/10/2013	(TXT) VERSION READY FOR THE PEER REVIEW
0.9	30/09/2013	(TXT) FINAL VERSION FOR FINAL CHECK
1.0	31/09/2013	(TXT) FINAL VERSION

DELIVERABLE PEER REVIEW SUMMARY

ID	**Comments**	**Addressed (✔) Answered (A)**
1	Sometimes you see "TELL ME" others "TELL-ME" other "TELLME". Please, just use the right one	✔
2	delete some pages in blank	✔
3	Update Index and Figures. Figures index some in bold.	✔
4	The acronym should be RWP or RW Platform; same for DW and AW	✔
5		

DELIVERABLE ACRONYMS

Acronym	**Full Name**
AR	Augmented Reality

LA	Learning Activity
LC	Learning Challenge
LCX	Learning Context
LCO	Learning COntents
LN	Learning Need
LP	Learning Path

TABLE OF CONTENTS

EXECUTIVE SUMMARY 8

1. INTRODUCTION 9

1.1 INTRODUCTION 9
1.2 APPLICABLE DOCUMENTS 10
1.3 POSITIONING OF WP2 IN TELL ME 11

2. INTRODUCTION TO WP2 COMPONENTS 12

2.1 OVERALL ARCHITECTURE AND WP2 12
2.2 WP2 COMPONENTS 13

3. REAL WORLD IOT PLATFORM COMPONENT 15

3.1 OVERALL DATA 15
3.2 ARCHITECTURE AND FUNCTIONALITIES 15
3.2.1 Introduction 15
3.2.2 Summary of Functionalities 15
3.2.3 Architecture 15
3.3 TECHNICAL INFORMATION 16
3.4 LICENSING 17
3.5 TECHNICAL MANUAL 17
3.5.1 IoTBroker installation 17
3.5.2 Esper installation 20
3.6 USER MANUAL 21
3.7 CONCLUSION AND FUTURE PLANS 21

4. DIGITAL WORLD IOS PLATFORM 22

4.1 OVERALL DATA 22
4.2 ARCHITECTURE AND FUNCTIONALITIES 22
4.2.1 Introduction 22
4.2.2 Summary of Functionalities 22
4.2.3 Architecture 22
4.3 TECHNICAL INFORMATION 23
4.3.1 Mashup Platform Composer 24
4.3.2 Request Widget 29
4.3.3 Activity Widget 38
4.4 LICENSING 45
4.5 TECHNICAL MANUAL 45
4.6 USER MANUAL 45
4.7 CONCLUSION AND FUTURE PLANS 45

5. VIRTUAL WORLD IOC PLATFORM 47

5.1 OVERALL DATA 47
5.2 ARCHITECTURE AND FUNCTIONALITIES 47
5.2.1 Introduction 47
5.2.2 Summary of Functionalities 47
5.2.3 Architecture 49
5.3 TECHNICAL INFORMATION 50

5.4 LICENSING 51
5.5 TECHNICAL MANUAL 52
5.6 USER MANUAL 52
5.7 CONCLUSION AND FUTURE PLANS 52

6. CONCLUSIONS AND NEXT STEPS 53

7. ANNEX A: MIDDLEWARE INTEGRATION LAYER (WD2.1 EXTRACT) 54

7.1 OVERALL DATA 54
7.2 ARCHITECTURE AND FUNCTIONALITIES 54
7.2.1 Introduction 54
7.2.2 Summary of Functionalities 54
7.2.3 Architecture 54
7.3 CONCLUSION AND FUTURE PLANS 55

8. ANNEX B: TELL ME ARCHITECTURAL DESIGN (WD6.1 EXTRACT) 56

8.1 INTRODUCTION 56
8.2 SYSTEM ACTORS AND USE-CASE DIAGRAM 56
8.3 ARCHITECTURE OVERVIEW 58
8.3.1 Logical Architecture (DOW) 58
8.3.2 First Architecture 58
8.3.3 Link to WPs implementing the architecture 60
8.3.4 Learning Contents Flow 61

List of Figures

Figure 1: TELL ME PERT Diagram and WP2 11
Figure 2: First version of the TELL ME architecture 12
Figure 3: Real World Platform 16
Figure 4 : The RabbitMQ Management plugin main interface. 18
Figure 5: Adding an exchange. 18
Figure 6: Adding an binding. 19
Figure 7: Adding an queues. 19
Figure 8: TELL ME configuration for RabbitMq. 20
Figure 9: Command to run CEP. 21
Figure 10. Architecture of DW IOS showing the conceptual interaction between trainee, R-WDGT, A-DGWT, mashup and MPC 23
Figure 11. Technical architecture of DW IOS 23
Figure 12. Conceptual view of information used/managed by the MPC 26
Figure 13. Request Widget (R-WDGT) conceptual mock-up 30
Figure 14. Generic widget/gadget architecture, with a REST WS back end. 30
Figure 15. Overall architecture encompassing W3C Packaged Web Apps (widget) front end (container) and back end (server) support 31
Figure 16. Example of deployment and widget structure in Tomcat + Wookie environment 33
Figure 17. Example of deployment and gadget structure in Tomcat + Shindig environment 34
Figure 18. Example of deployment in Apache Http Server environment 35
Figure 19. Request Widget (R-WDGT) UI – preconfigured learning needs (conceptual mock-up) 35
Figure 20. Request Widget (R-WDGT) UI – taxonomy tree viewer (conceptual mock-up) 36
Figure 21. Request Widget (R-WDGT) UI – text-based search (conceptual mock-up) 36
Figure 22. Request Widget (R-WDGT) UI (HTML5/CSS3/JS prototype) 37
Figure 23. Activity Widget (A-WDGT) conceptual mock-up 39
Figure 24. Activity Widget (A-WDGT) conceptual mock-up, showing the learning path partially consumed and the related trainee's feedback 39
Figure 25. Activity Widget (A-WDGT) conceptual mock-up, showing interaction/communication with other widget and possible logging approach 41
Figure 26. Activity Widget (A-WDGT) conceptual mock-up, details of the included feedback module associated with each step 41
Figure 27. Activity Widget (A-WDGT) conceptual mock-up 43
Figure 28. Activity Widget (A-WDGT) prototype (accordion model) 44
Figure 29. Activity Widget (A-WDGT) prototype (tree model) 44
Figure 30. Activity Widget (A-WDGT) prototype (tree model with drag&drop editing functionalities) 45
Figure 31 – VW IOC General Architecture 50
Figure 32: Internal Architecture of T2.4 55
Figure 33: TELL ME Actors **Error! Bookmark not defined.**

Figure 34: TELL ME use-cases diagram 57
Figure 35: TELL ME Architecture - DOW version 58
Figure 36: First version of the TELL ME architecture 59
Figure 37: TELL ME architecture – Learning Contents Flow 62

Executive summary

This document accompanies the release of the technical prototypes of WP2 DW-RW-VW platforms. In fact D2.1 is defined as "P" (prototype in the TELL ME DOW). In order to provide information about the work done in the WP and documentation supporting the understanding of the technical prototype, this document describes the specification of the WP2 components and their positioning as part of the overall TELL ME architecture. In particular the Real World, Digital World, and Virtual World Platforms are discussed.

- The **Real World Platform** comprises an IOT-based platform gathering data from the environment, managed by middleware and interpreted by a complex event processor that can launch new events and/or activate actuators. Events particularly interesting for TELL ME relate to the identification of learning needs about the worker;
- The **Digital World Platform** is based on a dynamic interactive graphic user interface and its back-end services that support the user in the management of the learning including: request, display, consumption and feedback of Learning Contents. The interface is based on a gadget environment;
- The **Virtual World Platforms** is based on a platform managing multimedia contents. From the repository to the end-user a professional streaming is offered on several devices. From the user to the repository the platform can ingest and tag videos produced by the user for further inclusion in the training material.

The three platforms will be completed and integrated by the fourth component called "**Privacy, Security and Platforms Interoperability Infrastructure**", due on M15 and composing the CORE of the TELL ME platform. The interim specification of this component extracted from working document 2.1 is reported as annex of this document. The platform is based on a set of SOA middleware products implementing: process management (BPEL) privacy and access control (Identity server) and mediation (ESB). The component will be extensively reported in D2.3 (M15 – prototype)

A second run of integration will take place at M15 between the before mentioned components and the ones implemented in WP3 (mostly related to the repository level) and WP4 (mostly related to the workplace and human-computer interaction). The design of the **overall architecture** is defined in WP6 reported in an internal document (WD6.1). In order to get the reader updated about how pieces fit together an extract to that deliverable is included as annex to this deliverable. The final version of the architecture will be released as project deliverable in M15 (D6.1 - Prototype).

1. Introduction

1.1 Introduction

WP2 is a technical workpackage of the TELL ME project; this document accompanies the technical prototype released at M12. First of all the WP2 positioning in the overall architecure is reported in chapter 2. In its details the document provides an overview of the Real World, Digital World, and Virtual World Platforms about the follwing items:

- General information about the architectural component
- the components internal architecture and major functionalities
- technical information about softwares used in the component (libraries, databases, etc.)
- licensing information regarding mainly third party software incorporated or used in the software
- technical manual in order to allow technicians to download, install, configure and run the system
- user manual supporting humans in accessing and using GUI based prototypes
- concusions and future plan.

The three platforms will be completed and integrated by the fourth component called "**Privacy, Security and Platforms Interoperability Infrastructure**" that will be delivered by M15 composing the CORE of the TELL ME platform. The interim specification of this component extracted from working document 2.1 is reported as annex of this document. The extensive report will be in D2.3 (M15 – prototype).

A second run of integration will take place at M15 between the before mentioned components and the ones implemented in WP3 (mostly related to the repository level) and WP4 (mostly related to the workplace and human-computer interaction). The design of the **overall architecture** is defined in WP6 reported in an internal document (WD6.1). In order to get the reader updated about how pieces fit together an extract to that deliverable is included as annex to this deliverable.

This document is intended to be read by both technicians and end-users.

1.2 Applicable Documents

List of applicable documents:

- TELL ME Description Of Work (DOW)
- TELL ME Working Document WD2.1 (Internal to the consortium)
- TELL ME Working Document WD6.1 TELL ME architecture - First Version Specification (Internal to the consortium)
- TELL ME Deliverable D3.1
- TELL ME Deliverable D4.1

1.3 Positioning of WP2 in TELL ME

TELL ME developments and iterations encompass: user requirements collection, architecture specification and implementation and software test and validation. As visible in the PERT diagram of the TELL ME project [DOW] WP2 receives in input the information about the TELL ME architecture from T6.1 (internal Working Document WD6.1) and implements/integrate its own components. The generated module will then be integrated and tested in the whole architecture in task 6.2 (first prototype – first iteration) and 6.3 (final prototype – second iteration).

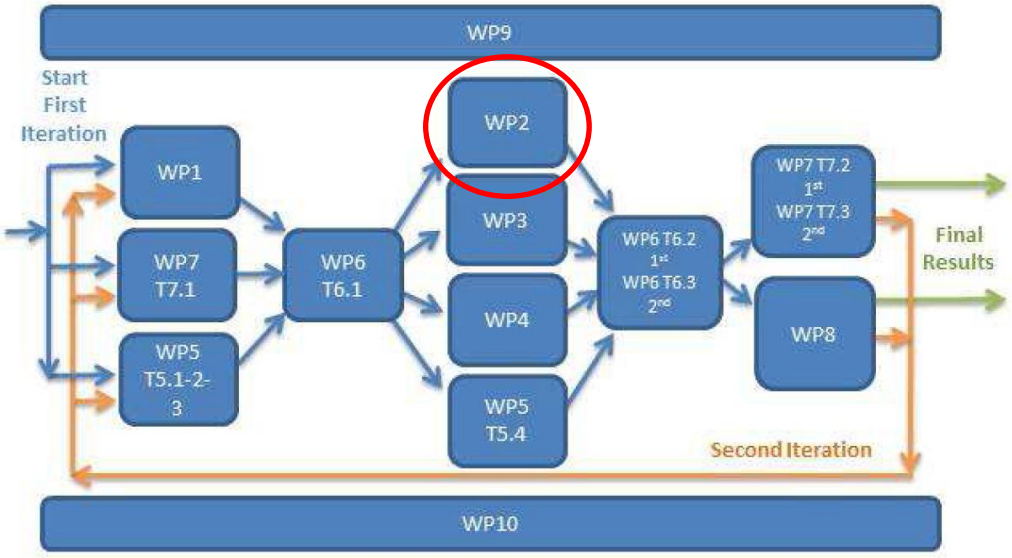

Figure 1: TELL ME PERT Diagram and WP2

2. Introduction to WP2 Components

2.1 Overall Architecture and WP2

In Figure 2 the technical architecture reported from the DOW is visible. The three platforms are represented in the centre of the architecture in orange managing the connection between the workplace and the repositories layer.

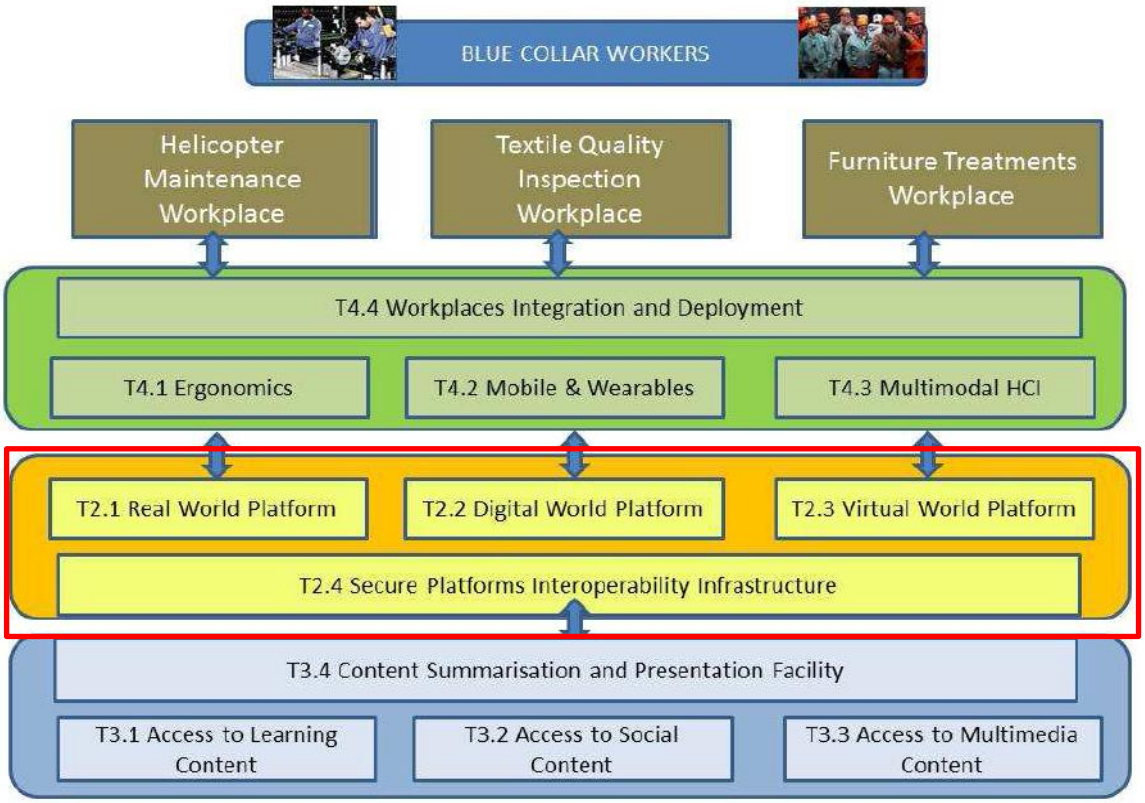
BLUE COLLAR WORKERS
Helicopter Maintenance Workplace
Textile Quality Inspection Workplace
Furniture Treatments Workplace
T4.4 Workplaces Integration and Deployment
T4.1 Ergonomics
T4.2 Mobile & Wearables
T4.3 Multimodal HCI
T2.1 Real World Platform
T2.2 Digital World Platform
T2.3 Virtual World Platform
T2.4 Secure Platforms Interoperability Infrastructure
T3.4 Content Summarisation and Presentation Facility
T3.1 Access to Learning Content
T3.2 Access to Social Content
T3.3 Access to Multimedia Content

Figure 2: First version of the TELL ME architecture

2.2 WP2 Components

Real World Platform (RWP): The goal of this component is to **monitor** the information coming from the smart workplace environment where the trainee is working in order to automatically **identify learning needs** (LNs) that should be sent to the decision making part of the architecture that will evaluate them and will provide an answer to the user. Information detected can come from various sources like machineries (eg: machinery error), environment (eg: temperature, humidity, light, noise sensors), person (eg: body movements, calories used) and in general any other source. Learning Needs will be sent to the CORE part of the TELL ME platform as ***enquiries*** following the eMeMO methodology.

The RW platform is composed of the following sub components that will be described more in details in chapter 3:

- IOT sources (sensors, etc)
- IOT Broker
- Complex Event Process (CEP)
- Actuators

Digital World IoS Platform (DWP): This platform aims at vehiculating human-generated requests and at bridging to the trainee the output coming out of the TELL ME decision-making components in order to satisfy the identified learning needs. On one side, this complements the RW platform (as for what concern learning need characterization) and on the other side encompasses all applicable scenarios for the delivery of the learning paths (or better "mixes") to the trainee in order to ease learning content consumption and support collection of feedbacks. The DW bundle includes three main components:

- **Request Widget (R-WDGT):** given the specific application scenario, the widget allows the trainee to autonomously express a learning needs by selecting predefined contextual learning paths (e.g. "mixes"), by selecting a set of meaningful tags from the prompted contextual taxonomies or by simply entering a free text search.
- **Activity Widget (A-WDGT):** the widget prompts to the trainee the learning path (i.e. "mix") suggested by the TELL Me platform and optimised to satisfy the given learning need, allows to move along the learning activities included and to launch the associated learning contents, giving alsothe possibility to rate/comment each item.
- **Mashup Platform Composer (MPC):** the component composes on the fly the UI in order to optimize the consumption of the returned mixes according to the given boundary conditions implicitly communicated by the context (e.g. available devices, location, etc.) or explicitly defined by the trainee while using the Request Widget.

Virtual World Platform IoC (VWP) The goal of this component is to provide a service platform for **Networked Media ingestion and channeling**. This includes media ingestion facilities for acquiring new audiovisual contents (e.g.: through mobile devices) to be properly transcoded and stored within the TELL-ME platform, making them available for further annotation through the media repository configuration interface provided in WP3. Moreover it

includes multi-channel delivery capabilities by matching the consumption device characteristics (e.g.: screen size, bandwidth, etc.) with the available media contents (e.g.: low definition video, high definition video, etc.). The VW platform functionalities described above are implemented through specific sub-components:

- Multi-Channel Streaming Engine, part of the platform delivery facilities;
- HTML5-based Web Player, focusing on the Web/Mobile channels;
- Media Ingestion component, related to images, videos and potentially network-based external sources (the different workplaces)

3. Real World IoT Platform

3.1 Overall data

Component Name	**RW IOT**
Reference WP – Task	WP2 – Task 2.1
Responsible	TXT
Version	1.0
Source Control	TELL ME SVN WP2 folder
Contact Person	Michele Sesana (michele.sesana@txtxgroup.com)
Short Description	This components monitors user activities and automatically discover Training Needs that are sent to the platform as *enquiries* following the eMeMO methodology.

3.2 Architecture and Functionalities

3.2.1 Introduction

The goal of this component is to **monitor** the information coming from the smart workplace environment where the trainee is working in order automatically **identify learning needs** (LNs) that should be sent to the decision making part of the architecture that will evaluate them and will provide an answer to the user. In principle, this component will also receive real-time data from the existing factory health-and-safety systems.

3.2.2 Summary of Functionalities

Major functionalities are:

- Gather events from smart objects
- Aggregate events by an IOT Broker
- Process events in real time
- Process events periodically
- Extract learning need
- Send commands to actuators

3.2.3 Architecture

Please find here below the fine grained architecture of the Real World IOT Platform.

The system is composed of IOT devices (sensors and actuators) depending on the specific workplace; a gateway to aggregate and dispatch events (including security issues) and a Complex Event Processor that is a component consuming events and, using business rules to identify training needs (for example the discovery of a specific event sequence). The output of the CEP component is a special event identifying a learning need that is sent to the reasoner to be processed (D2.3 will be released by M15).

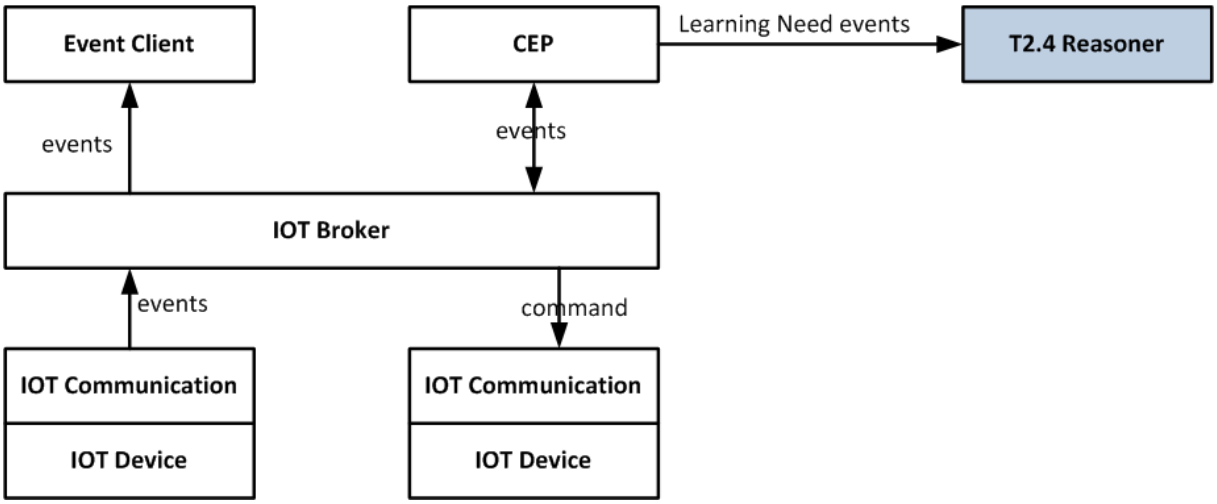

Figure 3: Real World Platform

3.3 Technical Information

In the following tables technical information about the architectural items are reported.

Name	IOT Broker
Nature	Middleware
Programming Language	Based on RabbitMQ
Development Tools	IOT broker online SDK
Additional libraries	
Application Server	apache
Databases	N/A

Name	Complex Event Processing (CEP)
Nature	event stream processing (ESP) and event correlation engine (CEP)
Programming Language	Based on Esper http://www.espertech.com/products/esper.php
Development Tools	Event Processing Language (EPL)
Additional libraries	
Application Server	apache
Databases	N/A

Name	Event Visualization
Nature	GUI
Programming Language	HTML5
Development Tools	Eclipse
Additional libraries	jQquery 1.10.2, jQuery Mobile 1.3.2, sockjs0.3
Application Server	Apache
Databases	N/A

The IOT communication device is completely dependent from the specific IOT device available in the environment of the end-users.

The reasoner will be implemented in T2.4 and released on M15.

3.4 Licensing

License for the RW platform is not yet determined.

Two types of software are used as the basis for this component:

- RabbitMQ is licensed under apache 2 licence
- Esper is licensed under GPL licence
- jQquery 1.10.2 is licensed under MIT license
- jQuery Mobile 1.3.2 is licensed under MIT license
- Sockjs0.3 is licensed under MIT license

3.5 Technical Manual

3.5.1 IoTBroker installation

Download and run the Erlang Windows Binary File (available on http://www.erlang.org/download.html). Then just run the installer, rabbitmq-server-3.1.5.exe (available on http://www.rabbitmq.com/download.html) and will set RabbitMQ up and running as a service, with a default configuration.

RabbitMQ provides an HTTP-based API for management and monitoring your RabbitMQ server, along with a browser-based UI and a command line tool.
The web UI is located at: http://server-name:15672/. To use the web UI you will need to authenticate as a RabbitMQ user (on a fresh installation the user "guest" is created with password "guest"). From here you can manage exchanges, queues, bindings, virtual hosts, users and permissions.

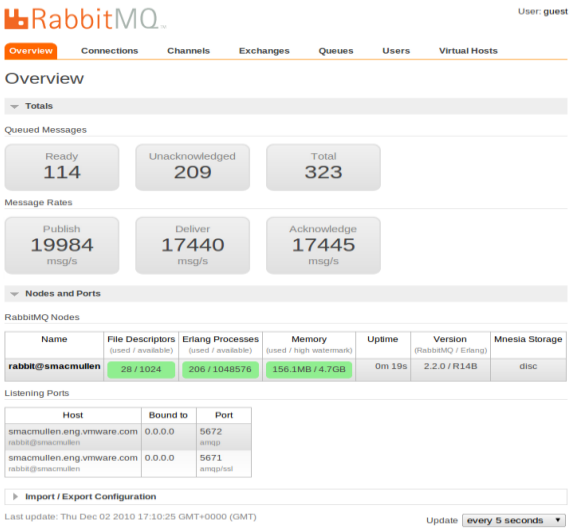

Figure 4 : The RabbitMQ Management plugin main interface.

3.5.1.1 Manage Exchanges

By Clicking on Exchanges on the navigation menu you'll get a list of the current exchanges available. In this list you can see the following information:

- Virtual host on which the exchange exists
- Exchange name
- Exchange type
- Parameters list;
- stands for durable for example;
- Message rate (in case you're publishing messages to this exchange)

To create a new exchange you should use the form under the Exchange list like show in the next figure.

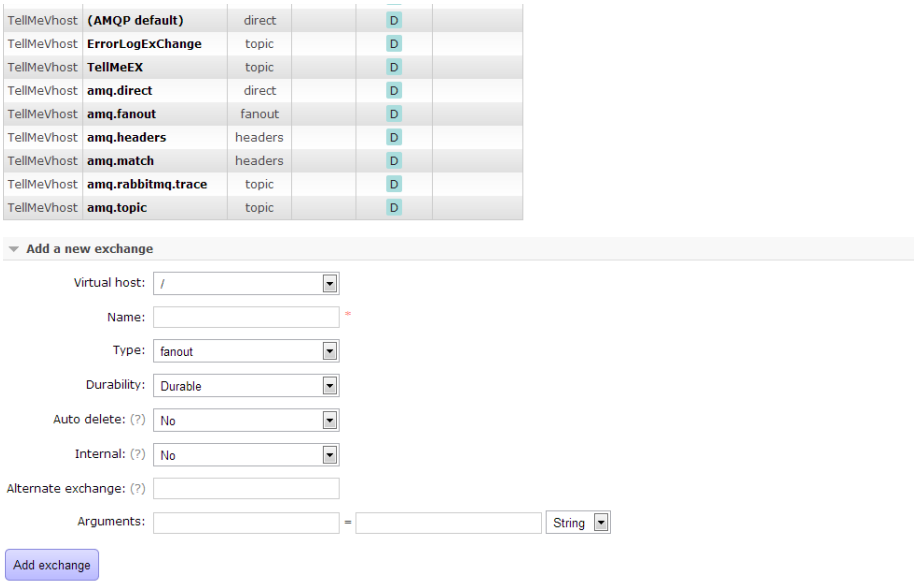

Figure 5: Adding an exchange.

By clicking on the exchange name, you can see more details about it, such as the exchange bindings (Exchange-Queues). You can also add new bindings and even delete the exchange. For create a new binding you must specify a queues name and rooting key used for routing message.

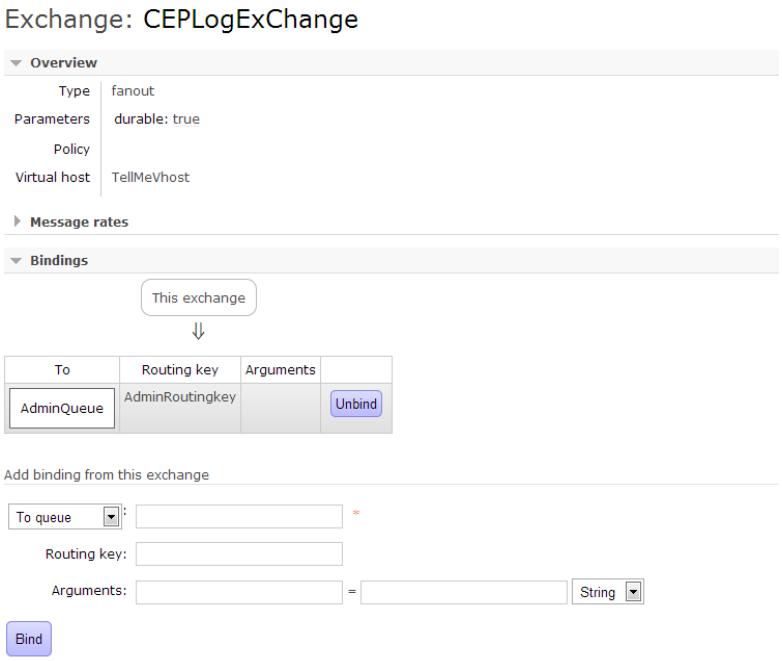

Figure 6: Adding a binding.

3.5.1.2 Manage Queues

Through queues tab you can manage the existing queues and create new ones. Information about the available queues are showed in the queue list that contains information like whether the queue is exclusive, the queue's status, and the queue's message rate. To create a new queue you should use the form under the queue list, like showed in the next figure.

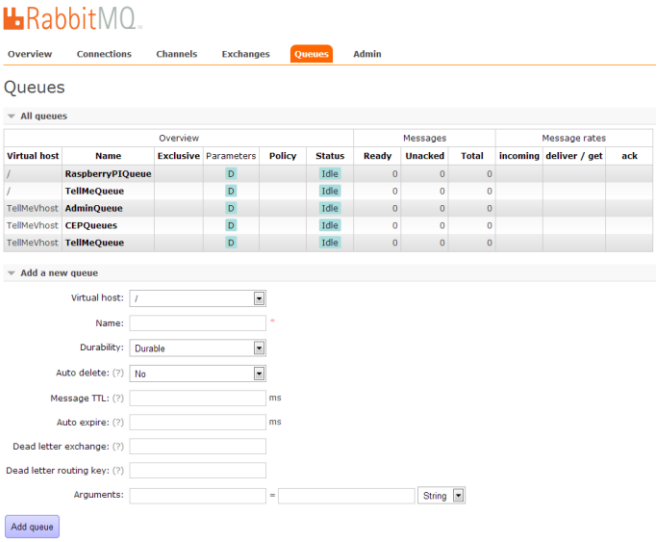

Figure 7: Adding an queues.

3.5.1.3 IoTBroker Tell Me configuration

In the Tell Me context RabbitMq must be config like shown in the next figure.

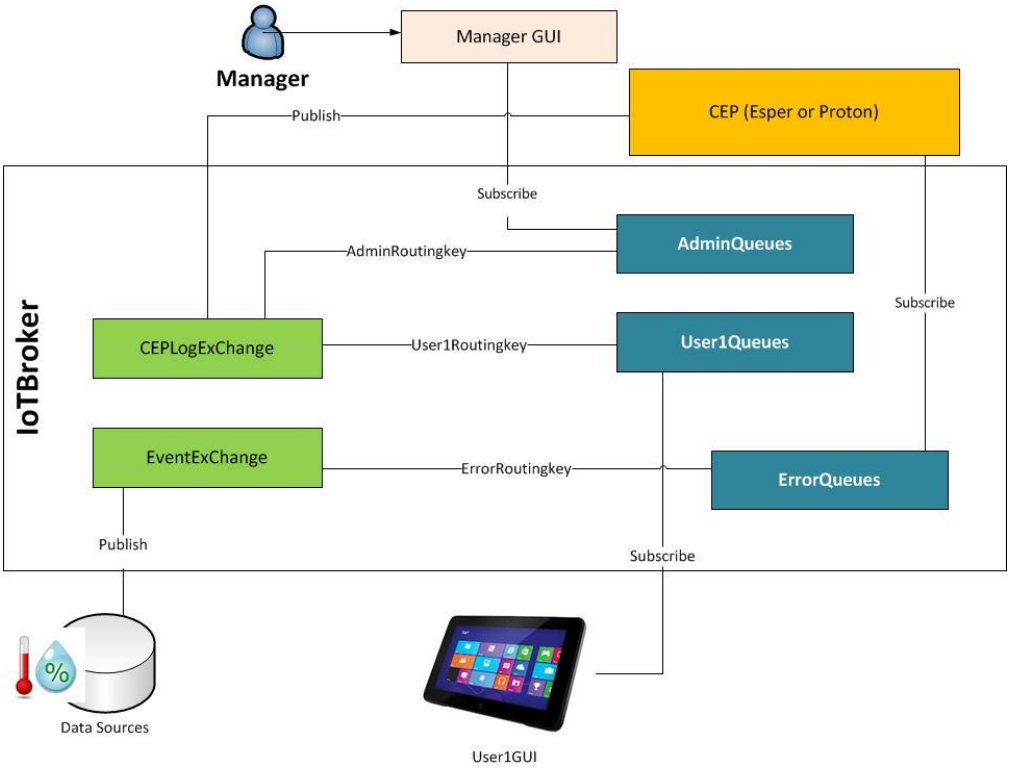

Figure 8: TELL ME configuration for RabbitMq.

EventExChange will collect data published from data sources that will be routed to *ErrorQueues*. CEP component is subscribed to *ErrorQueues* and publishes found error on *CEPLogExChange*. Lastly, the message in *CEPLogExChange* are routed to *AdminQueues* and *UserQueues* ready to be used by *UserGUI* and *ManagerGUI.*

3.5.2 Esper installation

Esper is easy to install and run: The first step is to download and unpack the distribution zip or tar file(available on http://esper.codehaus.org/esper/download/download.html). Provided you have a Java VM installed.

Esper consists of a jar file named "esper-version.jar" that can be found in the root directory of the distribution. Dependent libraries to Esper are in the "lib" folder. More information about Esper configuration and usage are available on http://esper.codehaus.org/tutorials/tutorial/tutorial.html.

In the TELL ME environment these libraries are used inside the CEP component. To start this componet is necessary execute TellMeCEP.jar available on TELL ME svn, using the following comand:

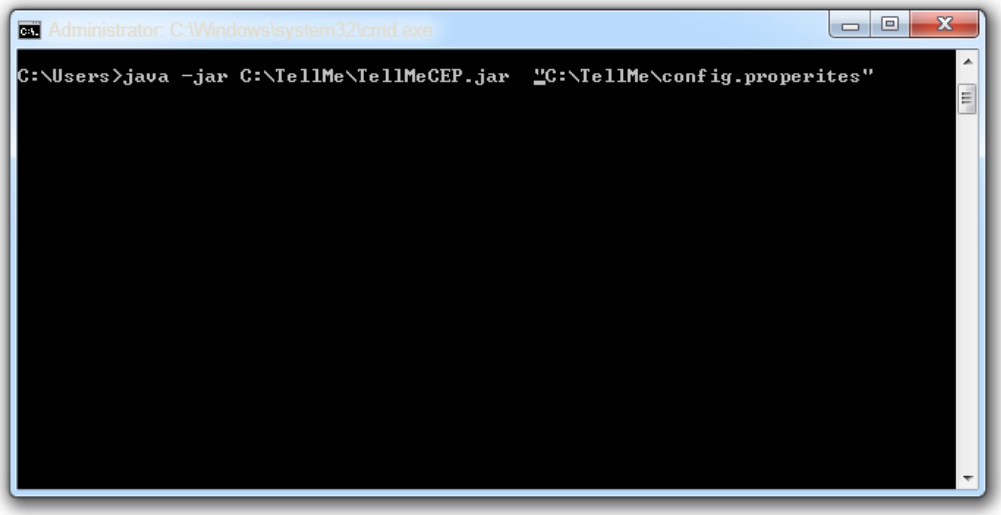

Figure 9: Command to run CEP.

The second parameter is the path of configuration file. In this file there are configuration parameter required for the communication between the CEP component and the IoTBroker.

3.6 User Manual

The user manual is not applicable for this component.

3.7 Conclusion and Future Plans

The system will be configured, instanciated and demonstrated as the first target in the AW scenario for the management of Foreign Object Damage. The platform will be integrated by M15 with the CORE part of the TELLLME system in order to manage the Learning Needs identified by the IOT system in the scope of WP2; with other architectural components in the scope of WP6 at M15 and with existing end-users systems in WP8.

4. Digital World IoS Platform

4.1 Overall data

Component Name	**DW IOS**
Reference WP – Task	WP2 – Task 2.2
Responsible	SOFT
Version	0.1
Source Control	TELLME SVN WP2 folder
Contact Person	Stefano Bianchi (stefano.bianchi@softeco.it)
Short Description	Digital World – Internet of Services

4.2 Architecture and Functionalities

4.2.1 Introduction

The DW IOS platform enables the delivery of basic TELL Me services allowing the trainee to interact with the system and be supported by the adaptation of learning paths ("mixes") and UI to boundary contextual information (workplace, expertise, devices etc.).

4.2.2 Summary of Functionalities

The platform is primarily meant to support:

- adaptation of the TELL ME system UI to the different potential containers for proper learning content delivery;
- customization of the trainee learning environment, by means of on-the-fly adaptation of the user interface on the basis of contextual information and of learning paths ("mixes") provided;
- specification of trainee's learning needs on the basis of specific requests (human input);
- visual delivery and tracking of guided consumption of learning paths ("mixes") and learning contents.

4.2.3 Architecture

The first prototype of the Digital World IoS platform includes three components:

1. **Mashup Platform Composer** (MPC)
2. **Activity Widget** (A-WDGT)
3. **Request Widget** (R-WDGT)

The MPC is a backend component which composes the mash-up for the trainees on the basis of the information received from the business logic components, whereas the A-WDGT and the R-WDGT are frontend components designed to interact with the user respectively in the

identification and specification of a learning need and in the tracking of the learning activities (consumption of a learning contents plus associated feedback) within a learning path.

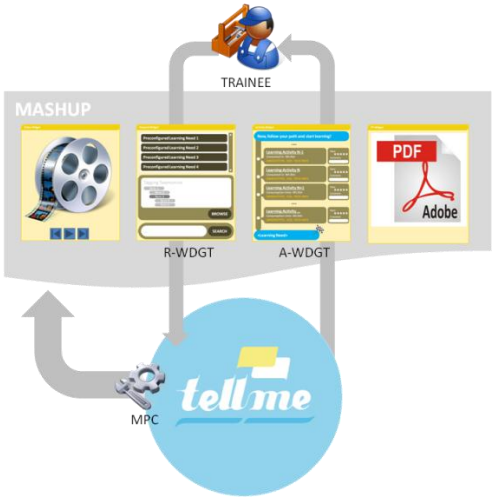

Figure 10. Architecture of DW IOS showing the conceptual interaction between trainee, R-WDGT, A-DGWT, mashup and MPC

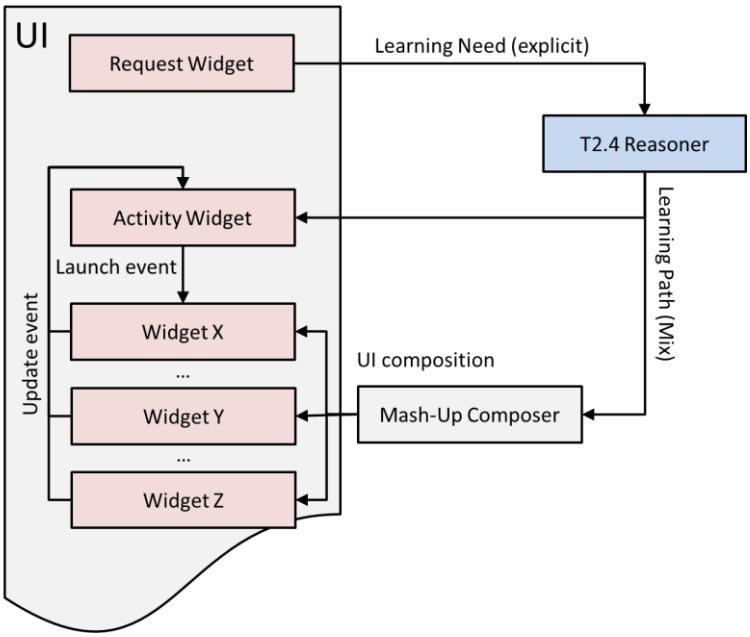

Figure 11. Technical architecture of DW IOS

4.3 Technical Information

The following sections provide detailed technical information on the single components belonging to the IoS platform.

4.3.1 Mashup Platform Composer

The Mash-up Platform Composer (MPC) is a backend component which composes the trainee's user interface "on the fly" on the basis of the information provided by the other back-end business logic components.

Following the approach adopted by several other previous research projects (see D1.1 "State of the Art in Learning and Creativity Methodologies in Manufacturing-related contexts" for complete references), TELL ME considers that the possibility to merge heterogeneous functionalities, tools and contents is a value added to enhance the learning processes, in particular for manufacturing environments. The overall project approach will ultimately deliver a mashup-based learning environment which will be personalized according to the user profile and expertise, contextualised by technological and workplace-related boundary constraints and fed up by learning contents to be consumed in several different ways (e.g. multimedia, AR, etc.).

4.3.1.1 Functionalities

The MPC supports the learning services at the workplace, providing generic capabilities and composition services and supporting re-use and evolution of single modules (e.g. HTML5 components, OpenSocial gadgets, or W3C-compliant widgets, a.k.a. Packaged Webapps) to address a wide range of learning situations and working environments (as exemplified by the three industrial application workplaces selected in the project).

The component acts as a proxy for the decision making components, and in particular for the learning method-based enquiry-Mix-experience-Match-Optimize (eMeMO) Rule Engine[1], that, given a context and a learning need, selects the Learning Contents (LCO) according to the eMeMO methodology and then aggregates them in specialised learning activity scripts within a broader learning path.

Once contextual information is sent to the TELL ME platform, i.e.:

- the type of activity currently performed by the worker (training, working eccetc.);
- the result of the activity workflow (training, working, eccetc.);
- the type of device used, to support media content adaptation;

the MPC supports the summarization and presentation functionalities, and in particular:

- receives the MMO-based learning path specifications, which include learning activity scripts, which include direct links to the learning contents to be delivered to the trainee;
- sends the MMO-based learning path specifications and the included information to the specific learning content delivery engine;

[1] The acronym MMO includes eMeMO (spoken as M-M-O): enquire-Mix-experience-Match-Optimise

- (optional, it can be "hard-coded" in the first version) queries the Widget Repository to select the most appropriate widgets for the delivery of contents on the basis of the scripts received by the Decision Making modules;
- configures the mash-up platform and compose the UI for the trainee by aggregating specific widgets, taking into account:
    - content adaptation (adapting web UI to mobile device);
    - geo-localisation of the device (if available or applicable);
    - localisation of the (wearable) device within the working place (if available or applicable);
    - available bandwidth (for media streaming optimization).

The MPC is closely coupled with the gadget/widget server/container that will host the different modules. According to the current design, the MPC is conceived as an extension of basic functionalities already provided by e.g. a generic widget/gadget engine, namely a server application that generates embeddable widget instances using HTML, CSS and JavaScript (by e.g. combining widget definitions extracted from a widget repository with widgets preferences derived from a widget preference server). The adoption and extension of e.g. an open source engine allows to exploit additional useful services such as, for example, proxy functionality for accessing remote services and support for authentication/authorisation.

4.3.1.2 Architecture

Information model

The MPC behaviour is dictated by the information received from the backend logic, which explicitly composes the learning contents to be consumed by the trainee. To illustrate the main concepts which model the whole process, it can be useful to follow the following schema:

- given that a Learning Need (LN) is defined as a triggering motivation to move between two Knowledge Statuses, differentiated by associated competences and skills, the TELL ME backend business logic components identifies on the basis of the MMO-methodology and of the Learning Context (LCX) (which includes user's profile and expertise, workplace characteristics, process information etc.) a personalized Learning Path;
- a Learning Path (LP) is composed by of different Learning Activities (LA), whose descriptions and granularity depend on the training methodologies and ontologies chosen to specify or support that specific LP, which identifies macro-areas of competence or skills to be gradually acquired along the learning process;
- each Learning Activity include a reference to the associated Learning Contents.

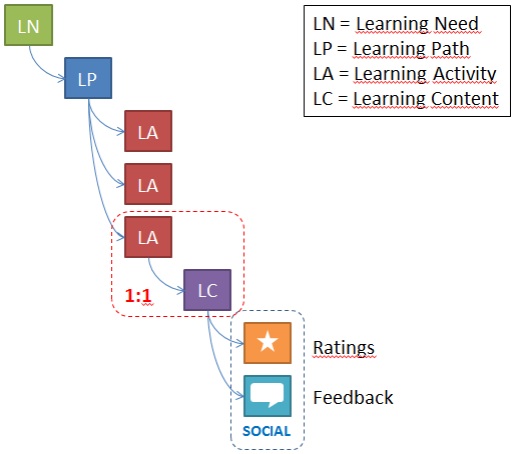

Figure 12. Conceptual view of information used/managed by the MPC

Information format

The MPC receives the description of an LP (which includes the links to the learning contents to be consumed, associated to/grouped within nested Learning Activities) in form of a simple JSON (JavaScript Object Notation) file.

The example below reports a LP which includes an array of LAs (only one LA included in the example) which includes an array of LCOs (only one LCO included in the example).

Each item has at least an id and a description in the language selected for the UI.

```
{
    "learningPath": {
        "id": "...",
        "description": "...",
        "author": {
            "id": "...",
            "name": "...",
            "surname": "...",
            "affiliation": {
                "id": "...",
                "name": "..."
            }
        },
        "learningActivityList": [
            {
                "learningActivity": {
                    "id": "...",
                    "description": "...",
                    "learningContentlist": [
                        {
                            "learningContent": {
                                "id": "...",
                                "description": "...",
                                "author": {
                                    "id": "...",
                                    "name": "...",
                                    "surname": "...",
                                    "affiliation": {
                                        "id": "...",
                                        "name": "..."
                                    }
                                }
                            }
                        }
                    ]
                }
            }
        ]
    }
}
```

The learning path script as defined above is agnostic as for the selection of the widgets to be used for the consumption of the given LCOs, as it simply declares what must be consumed along the specified learning process and does not containt explicit reference to the widget to be used and that must be included in the UI. The way the MPC will compose the UI by will be based on contextual information (e.g. the available device on which contents will be consumed) and on an explicit mapping between categories/types of contents (e.g. media, AR, etc.) and associated engines, whose frontend will be generally implemented as a widget/gadget/HTML5 fragment.

An explicit declarative mapping between content categories/types and widget/gadget will be used by the backend components to generate additional attributes for the learning content element (such as e.g. “engine” or “widget”) or by the MPC itself to invoke on the fly the mapped engines/widgets passing the link to the content to be consumed.

Integration

As previously illustrated, the MPC component acts “behind the scene” to render the trainee’s user interfaceUI in compliance with the generated eMeMO-based learning path and technical boundary constraints (e.g. devices, local ergonomic considerations such as noise levels). This role implies an integration within the TELL ME system somewhere between the rendering/visualization layer and the backend components, with different approaches related to overall development methodology. Two main alternatives are currently considered for the development activities, according to the use of OpenSocial Gadget / W3C Packaged Webapp (widget) technologies or the development of an ad-hoc light container:

- extension and customization of native functionalities provided by an (open source) widget/gadget server/container;
- development of an HTML5/CSS3/JS-based modular RIA following a Model-View-Controller (MVC) pattern where the MPC composed the UI selecting the available Views and instantiating the associated controllers.

4.3.1.3 Technical Information

Nature	*DW IOS Mashup Platform Composer, business logic/backend*
Programming Language	*Java, XML*
Development Tools	*Eclipse IDE*
Additional libraries	*Java XML parsing libraries* *Java JSON parsing libraries*
Application Server	*Apache Tomcat (http://tomcat.apache.org)* *Apache Shindig (http://shindig.apache.org/) is an OpenSocial container and helps start hosting OpenSocial apps quickly by providing*

	the code to render gadgets, proxy requests, and handle REST and RPC requests. *ROLE SDK (http://www.role-project.eu/Software-Development)*
Databases	*See WP3 repository components for common technical details*

4.3.1.4 Licensing

According to the two different integration modalities illustrated in the previous section, the MPC will be released with at least two distinct licenses (being the MPC implemented in a dual way):

- In case of a proprietary solution (HTML5/CSS3/JS framework), the MPC will be available under a proprietary commercial license, IPR being owned by Softeco Sismat Srl only;
- In case of a solution extending core functionalities of available OS containers (such as Apache Shindig or ROLE SDK, mentioned in the table), the applicable OS license will be accordingly applied (Apache Shindig is e.g. released under the terms of the Apache Software License, Version 2.0).

4.3.1.5 Technical Manual

As an extension of native libraries of an OS container, the MPC could require recompiling source code to customize the default behaviour of the container, therefore specific instructions for the container could apply case-by-case. As for the implementation of the MPC as a module of an HTML5 AJAX RIA application, according to the framework adopted (e.g. jQuery, DOJO, etc.), slightly different installation and deployment guidelines apply in compliance with the related libraries. In case of pure client-side applications, the MPC logic is built on Javascript libraries which dinamically compose the page including the necessary modules and HTML fragments. In case of AJAX-based server side applications (whose UI is rendered by client Javascript engine as HTML5), the MPC logic is built on Java classes which model and resolve the association between content types and widgets/gadget, interpreting the LP model received and composing the UI with AJAX push response from the server.

4.3.1.6 User Manual

Not applicable as the MPC component is not directly used by the TELL ME users.

4.3.1.7 Conclusion and Future Plans

Not applicable at the current stage of design/development.

4.3.2 Request Widget

The Request Widget (shortly R-WDGT) is a GUI used by a blue collar worker (a.k.a. trainee) or a training manager to express a learning need and get support from the TELL ME system in terms of a personalised, customized and optimised learning path.

The R-WDGT is the entry point for the interaction of the user with the system, allowing the user to notify TELL ME components with an explicit request formalized after the detection of a knowledge gap within the workplace.

As the overall content management approach includes the adoption of domain-specific taxonomies for tagging the available learning contents, the widget includes also basic taxonomy browsing and searching features which are basic yet usually appreciated functionalities. By using browsing and search functionalities, the system simply acts as a learning content delivery system: learning contents will be proposed according to the user's profile and the detected technical constraints.

In principle the request widget can be extended to support tasks in content development and in evaluating a proposed learning path. Possible extensions to requests include: 'nearest match' (between training needs and locally-available learning contents); and 'goal seek' (federated search of other sources of learning contents, to discover content elsewhere that could be re-purposed to meet the local need).

4.3.2.1 Functionalities

As the TELL ME system is meant to provide support to blue collar workers during training activities within the working environment, a certain degree of customization is necessary for the final deployment, as demonstrated by the implementation of the three different pilots.

Given that each working environment is characterized by different features and processes (general e.g. health and safety at work, security and accident prevention and management or specific activity-related topics), it is likely to that – given both general (e.g. health and safety at work, security and accident prevention and management) or specific activity-related topics – it is possible to identify for each specific application some preconfigured learning needs, to be possibly mapped onto e.g. the type of activity performed or the professional figure goals involved.

The LP returned by the system and designed according to the MMO methodology will be then customized by the system according to the user profile (previous expertise, provided feedback, etc.) and the contextual information.

The response given by the system to the request forwarded by the widget is visualized in form of a learning path (if applicable, once the system react to a click on a preconfigured learning need) or as a plain list of LCOs (once the system retrieves learning contents according to a taxonomy-based browsing or a searching command) within the Activity Widget (A-WDGT).

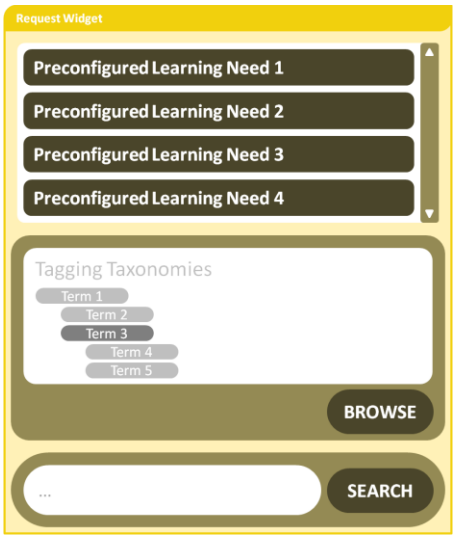

Figure 13. Request Widget (R-WDGT) conceptual mock-up

4.3.2.2 Architecture

As a general vertical development approach, both widgets and gadgets supports the possibility to have a light yet functionally rich HTML5/CSS3/JS-based client on the browser and a back end logic provided by e.g. remote web services interacting with legacy repositories etc.

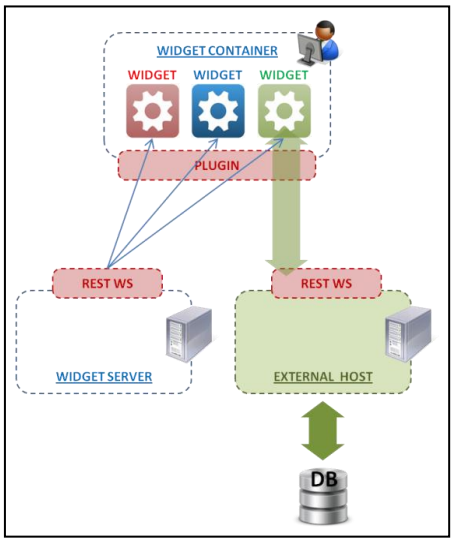

Figure 14. Generic widget/gadget architecture, with a REST WS back end.

The same overall approach might have slightly different implementation and deployment flavours, provided that widget-based or gadget-based mash-ups look and behave the same.

The following figure illustrates the architecture for a widget-based mashup built on top of a W3C Packaged Web Apps (widget) approach. In this case, a Widget Server for deployment and a Widget Container for consumption are included, with several different options.

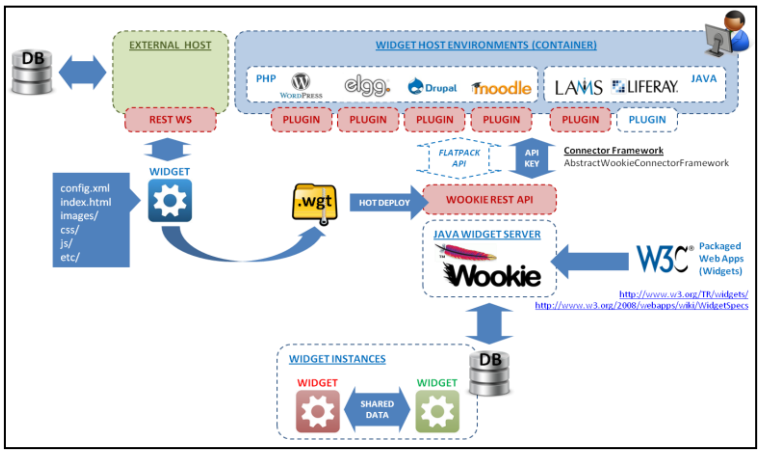

Figure 15. Overall architecture encompassing W3C Packaged Web Apps (widget) front end (container) and back end (server) support

4.3.2.3 Technical Information

Nature	*User interface widget/gadget, frontend*
Programming Language	*HTML5, CSS3, JavaScript + Java based Web Service backend*
Development Tools	*Eclipse IDE*
Additional libraries	*The following JavaScript libraries and presentation frameworks are candidates under evaluation:* *jQuery (http://jquery.com/) is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.* *jQueryUI(http://jqueryui.com/)* *jQueryMobile (http://jquerymobile.com/)* *DOJO (http://dojotoolkit.org/) is an open source modular JavaScript library (or more specifically JavaScript toolkit) designed to ease the rapid development of cross-platform, JavaScript/Ajax-based applications and web sites.* *Web service support:* *Apache Tomcat (http://tomcat.apache.org) is an open source software implementation of the Java Servlet and JavaServer Pages technologies.* *Apache AXIS (http://axis.apache.org/) is a Web Services / SOAP / WSDL engine, the successor to the widely used Apache Axis SOAP stack.* *Apache CXF (http://cxf.apache.org/) is an open source services*

	framework that helps build and develop services using frontend programming APIs, like JAX-WS and JAX-RS. These services can speak a variety of protocols such as SOAP, XML/HTTP, RESTful HTTP, or CORBA and work over a variety of transports such as HTTP, JMS or JBI.
Application Server	*Apache Tomcat (http://tomcat.apache.org)* *Apache Shindig (http://shindig.apache.org/) is an OpenSocial container and helps start hosting OpenSocial apps quickly by providing the code to render gadgets, proxy requests, and handle REST and RPC requests.* *ROLE SDK (*http://www.role-project.eu/Software-Development*)*
Databases	*See WP3 repository components for common technical details*

4.3.2.4 Licensing

Since the R-WDGT is a pivotal component of the TELL ME system UI and that the correct use of the TEL functionalities is enabled by the expression of a learning need ("Enquire" phase of the overall eMeMO methodology), the R-WDGT will be released under the same OS license that will be applied to the TELL ME Core components.

4.3.2.5 Technical Manual

Provided that the widget can be implemented with the same functionality as a W3C Packaged Webapp (widget) or as an OpenSocial gadget, starting from a common HTML5/CSS3/JS base, the following sections summarize generic structural and deployment information and provide reference to official documentation.

Widgets

According to the W3C Packaged webapps specifications[2]:

> *A widget package is a valid Zip archive that contains the following:*
>
> - *One or more start files, located at the root of the widget package and/or at the root of locale folders or referenced by a content element's src attribute.*
> - *One configuration document, located at the root of the widget package.*
> - *Zero or more icons, either located at the root of the widget package and/or at the root of locale folders or referenced by the icon element's src attribute.*
> - *Zero or more arbitrary files located either at the root of the widget package and/or in arbitrary folders or in locale folders.*
> - *Zero or more digital signatures located at the root of the widget package.*

[2] http://www.w3.org/TR/widgets/

A typical configuration file example is:

```
<?xml version="1.0" encoding="UTF-8"?>
<widget xmlns       = "http://www.w3.org/ns/widgets"
        id          = "http://www.tellme-ip.eu/widgets/exampleWidget"
        version     = "1.0"
        height      = "400"
        width       = "400"
        viewmodes   = "fullscreen">

  <name short="Example 2.0">
    The example TELL ME widget!
  </name>

  <description>
    A sample TELL ME widget!
  </description>

  <author href  = "http://www.tellme-ip.eu/"
          email = "consortium@tellme-ip.eu">TELL ME Consortium</author>

  <icon src="icon.png"/>

  <content src="index.html"/>

  <license>
  ...
  </license>
</widget>
```

Example of deployment within the Tomcat Servlet Container using the Apache Wookie Widget server (http://wookie.apache.org) :

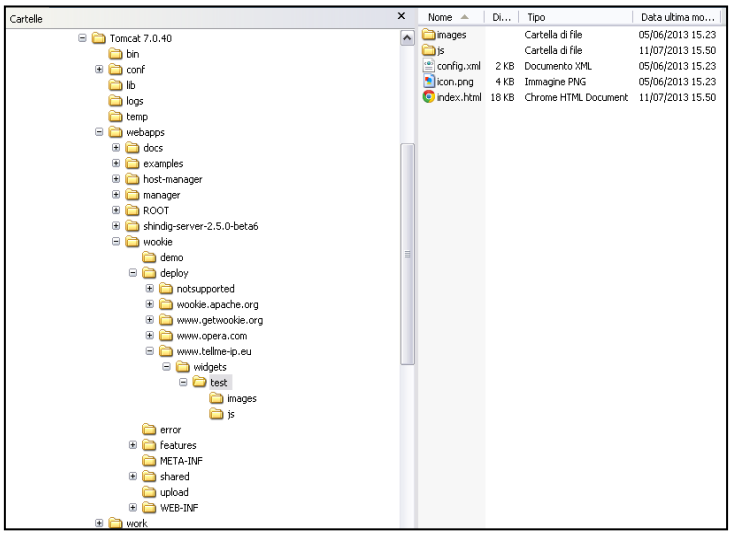

Figure 16. Example of deployment and widget structure in Tomcat + Wookie environment

Gadgets

In compliance with the specification from the OpenSocial alliance, the `TellMeTestGadget.xml` definition file for a basic "Hello TELL ME!" gadget appears like the following excerpt:

```
<?xml version="1.0" encoding="UTF-8"?>
<Module>
```

```
  <ModulePrefs title="TELL ME Gadget"
               description="TELL ME Test Gadget"
               thumbnail="http://www.tellme-ip.eu/thumbnails/tumbnail.png"
               icon="http://www.tellme-ip.eu/icons/icon.png">
  </ModulePrefs>
  <Content type="html">
    <![CDATA[

    <html>
        <head>
            <script src="....js"></script>
        </head>
        <body>
            <div id="...">Hello TELL ME!</div>
        </body>
    </html>

  ]]>
  </Content>
</Module>
```

Example of deployment within the Tomcat Servlet Container using the Apache Shindig Gadget container (http://shindig.apache.org) :

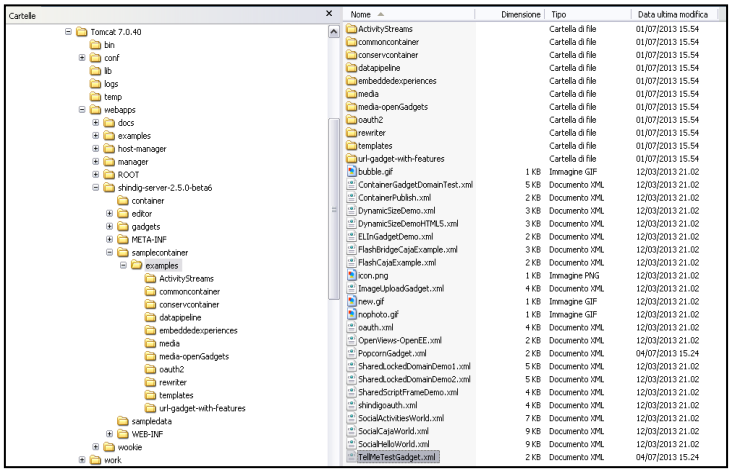

Figure 17. Example of deployment and gadget structure in Tomcat + Shindig environment

HTML5/CSS3/JS

For the deployment of the components as plain HTML5/CSS3/JS software, it is sufficient to publish them on a http server such as Apache HTTP server[3].

[3] http://httpd.apache.org/

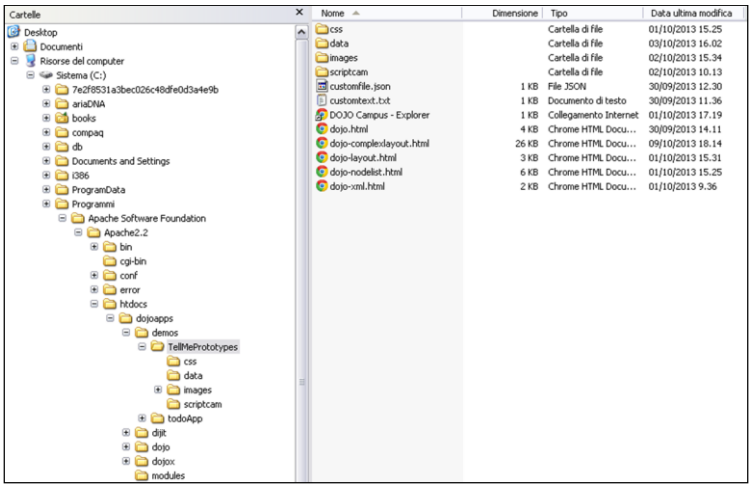

Figure 18. Example of deployment in Apache Http Server environment

The screenshots of the prototypes included in this deliverable has been taken from the HTML5/CSS3/JS UI deployed under the Apache Tomcat Servlet container, whose functionalities for plain html file provision can be exploited as well as in Apache HTTP server.

4.3.2.6 User Manual

The R-WDGT UI is composed by three main parts:

- a viewer for preconfigured learning needs associated to the trainee's profile and contextual information;
- a taxonomy viewer for the selection of domain tags related to the learning need;
- a simple text-based search with auto-completion operated on available tags.

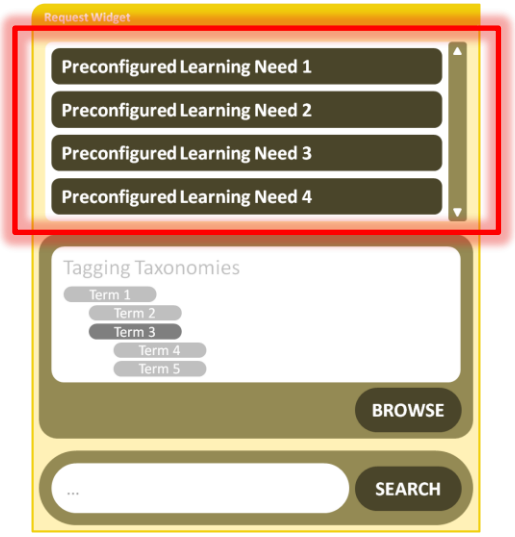

Figure 19. Request Widget (R-WDGT) UI – preconfigured learning needs (conceptual mock-up)

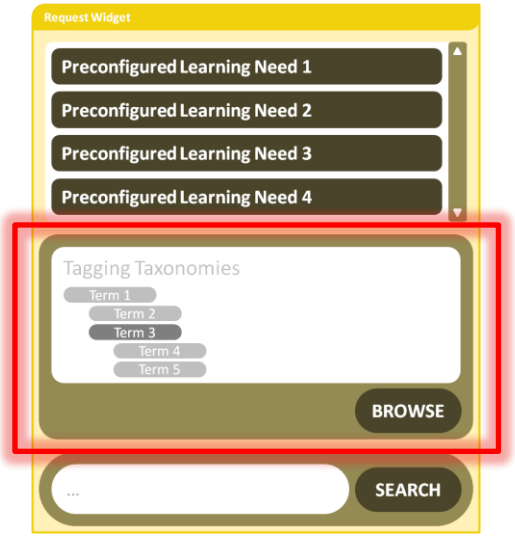

Figure 20. Request Widget (R-WDGT) UI – taxonomy tree viewer (conceptual mock-up)

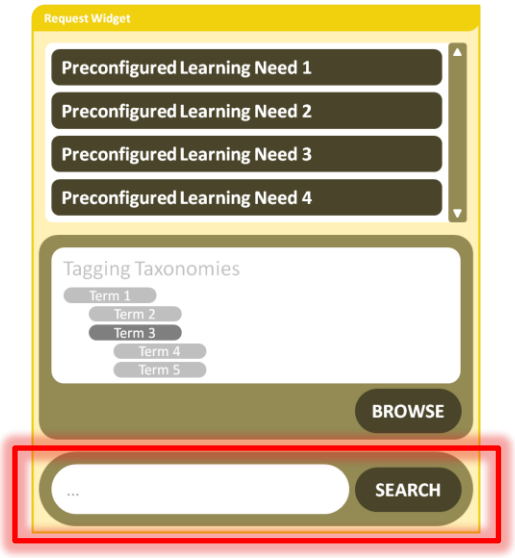

Figure 21. Request Widget (R-WDGT) UI – text-based search (conceptual mock-up)

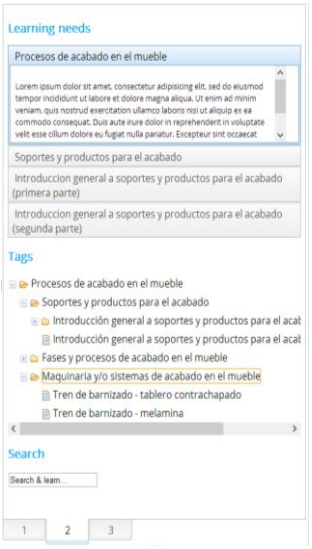
Learning needs
Procesos de acabado en el mueble
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat
Soportes y productos para el acabado
Introduccion general a soportes y productos para el acabado (primera parte)
Introduccion general a soportes y productos para el acabado (segunda parte)
Tags
Procesos de acabado en el mueble
Soportes y productos para el acabado
Introducción general a soportes y productos para el acat
Introducción general a soportes y productos para el acat
Fases y procesos de acabado en el mueble
Maquinaria y/o sistemas de acabado en el mueble
Tren de barnizado - tablero contrachapado
Tren de barnizado - melamina
Search
Search & learn...
1
2
3

Figure 22. Request Widget (R-WDGT) UI (HTML5/CSS3/JS prototype)

By clicking on a preconfigured learning need, the trainee requests to the system the optimised mixes of learning contents corresponding to the implicit learning requirements. The request corresponds to a statement like "I need to learn about {tag_i, …, tag_j, …., tag_k}", the set of tags being combined by e.g. trainer or training manager in a preliminary configuration phase.

By selecting a set of items in the prompted taxonomies, the trainee requests to the system the optimised mixes of learning contents corresponding to the explicit learning requirement. The request substantially corresponds to "I need to learn about {tag_i, …, tag_j, …., tag_k}", the set of tags being componed on the fly, which is answered by a mix encompassing all selected tags, according to the annotation of the contents and the results of the matching algorithms.

By entering a free text search string, the trainee is prompted with matching terms belonging to the taxonomy trees associated to the applicable worplace domain. In this modality, the search is a quick way to access the taxonomy trees without browsing. If no matching terms are prompted and selected, the search is performed directly on information available for the title and description of learning paths, learning activities, learning contents. The response provided by the system is composed by the learning paths which are associated to the matching items, optimised according to the backend algorithms.

4.3.2.7 Conclusion and Future Plans

In principle, the request widget can be extended to support tasks also in content development and in evaluating a proposed learning path. Possible extensions to requests include:

- "nearest match" (between training needs and locally-available learning contents);

- "goal seek" (federated search of other sources of learning contents, to discover content elsewhere that could be re-purposed to meet the local need).

4.3.3 Activity Widget

The Activity Widget (A-WDGT) prompts to the user a visual interactive representation of the optimised LP as designed and espressed expressed by the eMeMO Rules Engine and then used by the MPC to compose the trainee's UI.

The widget allows the trainee to have an immediate feedback on the consumption of the LCOs mixed in the LP, as well as providing the entry point to launch the different widgets with the associated LCOs.

Each invoked widget should be able to track somehow the consumption of the given LCOs so to allow the A-WDGT to change state accordingly.

The A-WDGT includes also a simple feedback UI to allow the trainee to express an evaluation of the LA and of the LCOs in terms of e.g. a vote or a comment, to feed the community components.

4.3.3.1 Functionalities

Learning path visualization and interaction

The following figures illustrate the A-WDGT conceptual mockup showing a LP depicted as a series of LAs, each pointing to a LCO (one-to-many relation applies all over for general design).

Figure 23. Activity Widget (A-WDGT) conceptual mock-up

Clicking on a single LA would allow the trainee to consume the associated LCO in the matching widget, marking as "done" the LA once the consumption is complete. The ongoing activity is highlighted as well.

Figure 24. Activity Widget (A-WDGT) conceptual mock-up,
showing the learning path partially consumed and the related trainee's feedback

Activity (experience) logging

In order to follow the latest advancements in the learning domain, the TELL ME project is considering to adopt the TinCan-Experience API recently issued by the ADL – Advanced Distributed Learning Initiative (http://tincanapi.com). As reported in the latest version of the specifications (v1.0):

> *The Experience API is a service that allows for **statements of experience** to be delivered to and stored securely in a Learning Record Store (LRS). These statements of experience are typically **learning experiences**, but the API can address statements of any kind of experience.*

A single experience is formalized and tracked using a statement,

> *A simple construct consisting of <Actor (learner)> <verb> <object>, with <result>, in <context> to track an aspect of a learning experience. A set of several Statements may be used to track complete details about a learning experience.*

and can be easily expressed in JSON like:

```
{
"id": "12345678-1234-5678-1234-567812345678",
```

```
"actor":{
    "mbox":"mailto:xapi@adlnet.gov"
},
"verb":{
    "id":"http://adlnet.gov/expapi/verbs/created",
    "display":{ "en-US":"created" }
},
"object":{
    "id":"http://example.adlnet.gov/xapi/example/activity"
}
}
```

Statements have several properties: <Actor>, <Verb>, and <Object> are required, whereas all other properties are optional:

- <result> (Object Result Object): further details representing a measured outcome relevant to the specified Verb.
- <context> (Object Context): gives the Statement more meaning.
- Examples: a team the Actor is working with, altitude at which a scenario was attempted in a flight simulator.
- <timestamp> (Date/Time): Timestamp (Formatted according to ISO 8601) of when the events described within this Statement occurred.
- <stored> (Date/Time): Timestamp (Formatted according to ISO 8601) when this Statement was recorded.
- <authority> (Object): Agent who is asserting this Statement is true.
- <version> (Version) The Statement‟s associated xAPI version, formatted according to Semantic Versioning 1.0.0
- <attachments> (Array of attachment Objects): Headers for attachments to the Statement.

It is interesting to notice that the Experience API allows tracking several different types of information, e.g. completions, scores, questions, bookmarks etc., thus supporting the tracking of different yet complementary types of information.

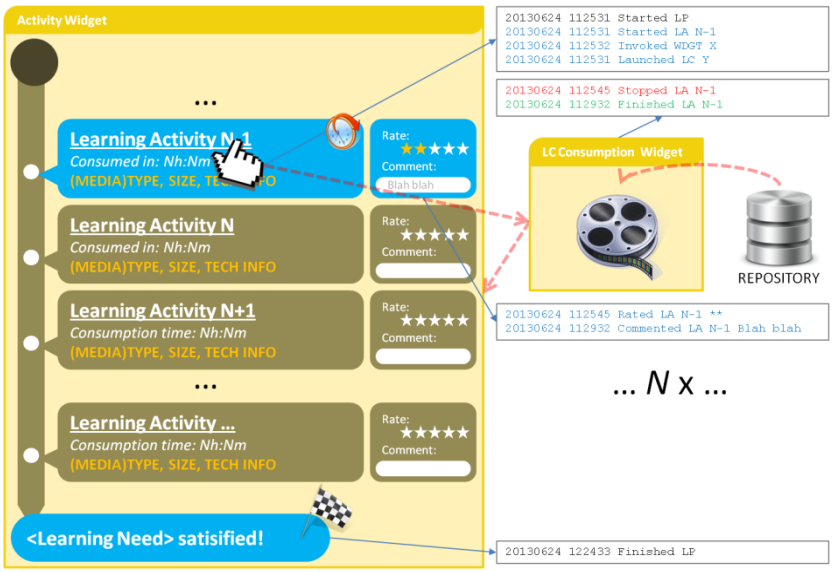

Figure 25. Activity Widget (A-WDGT) conceptual mock-up, showing interaction/communication with other widget and possible logging approach

The A-WDGT should thenwill track the trainee's activity by sending back to the system for proper storage and later retrieval the associated statement for proper storage, analysis and later retrieval. An interaction between the active widget consuming the LCO and the A-WDGT should must be implemented (e.g. event notification) to allow a proper logging of timing and results of the actions performed.

(Social) Feedback

In order to support social-oriented functionalities, the Activity WidgetA-WDGT includes a simple functionality which allows the trainee to rate the activity carried out (to which a learning content is associated) and to leave a comment about it. To provide a more granular management of information, the functionality might include the possibility to set the visibility of the comment expressed to one from a predefined set of addressees (e.g. trainer/training manager, community, colleagues etc.) as defined for the specific working environment and the associated policies.

Figure 26. Activity Widget (A-WDGT) conceptual mock-up, details of the included feedback module associated with each step

4.3.3.2 Architecture

See analogue section of the Request Widget.

4.3.3.3 Technical Information

Nature	*User interface widget/gadget, frontend*
Programming Language	*HTML5, CSS3, JavaScript + Java based Web Service backend*
Development Tools	*Eclipse IDE*
Additional libraries	*See Activity Widget details*

Application Server	*See Activity Widget details*
Databases	*See WP3 repository components for details*
Nature	*User interface widget/gadget, frontend*
Programming Language	*HTML5, CSS3, JavaScript + Java based Web Service backend*
Development Tools	*Eclipse IDE*
Additional libraries	*jQuery (http://jquery.com/) is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.* *jQueryUI(http://jqueryui.com/)* *jQueryMobile (http://jquerymobile.com/)* *DOJO (http://dojotoolkit.org/) is an open source modular JavaScript library (or more specifically JavaScript toolkit) designed to ease the rapid development of cross-platform, JavaScript/Ajax-based applications and web sites.* *Web service support:* *Apache Tomcat (http://tomcat.apache.org) is an open source software implementation of the Java Servlet and JavaServer Pages technologies.* *Apache AXIS (http://axis.apache.org/) is a Web Services / SOAP / WSDL engine, the successor to the widely used Apache Axis SOAP stack.* *Apache CXF (http://cxf.apache.org/) is an open source services framework that helps build and develop services using frontend programming APIs, like JAX-WS and JAX-RS. These services can speak a variety of protocols such as SOAP, XML/HTTP, RESTful HTTP, or CORBA and work over a variety of transports such as HTTP, JMS or JBI.*
Application Server	*Apache Tomcat (http://tomcat.apache.org)* *Apache Shindig (http://shindig.apache.org/) is an OpenSocial container and helps start hosting OpenSocial apps quickly by providing the code to render gadgets, proxy requests, and handle REST and RPC requests.* *ROLE SDK (*http://www.role-project.eu/Software-Development*)*
Databases	*See WP3 repository components for common technical details*

4.3.3.4 Licensing

Since the A-WDGT is a pivotal component of the TELL ME system UI and that the correct use of the TEL functionalities is enabled by the consumption of a Learning Path / mix ("Experience" phase of the overall eMeMO methodology), the A-WDGT will be released under the same OS license that will be applied to the TELL ME Core components.

4.3.3.5 Technical Manual

See analogue section for the Request Widget.

4.3.3.6 User Manual

The A-WDGT prompts the trainee with a learning path visualized as a series of ordered learning activities each having at least one associated learning content.

The trainee can click on the activities launching the included learning content in the associated widget/gadget.

Figure 27. Activity Widget (A-WDGT) conceptual mock-up

Once the learning content has been consumed, upon automatic notification from the running widget/gadget or upon explicit action by the trainee, the activity is marked as completed, and the trainee can also express a rating and a comment on the consumed learning content.

The conceptual mock-up, which has been generally validated to support the Experience step in the eMeMO methodology (the A-WDGT simply acts as a hook for the experiences being consumed in the different widgets, according to the learning path/mix generated by the system), has been prototyped in two different flavours, to allow evaluating different approaches from the point of view of usability:

- Accordion model
- Tree model

The same data model is shared among the two implementations, which render information as the view part of the renown Model-View-Control pattern.

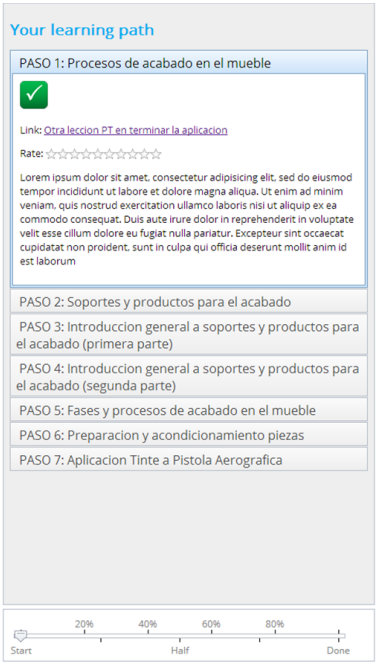
Your learning path
PASO 1: Procesos de acabado en el mueble
Link: Otra leccion PT en terminar la aplicacion
Rate:
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum
PASO 2: Soportes y productos para el acabado
PASO 3: Introduccion general a soportes y productos para el acabado (primera parte)
PASO 4: Introduccion general a soportes y productos para el acabado (segunda parte)
PASO 5: Fases y procesos de acabado en el mueble
PASO 6: Preparacion y acondicionamiento piezas
PASO 7: Aplicacion Tinte a Pistola Aerografica
20%
40%
60%
80%
Start
Half
Done

Figure 28. Activity Widget (A-WDGT) prototype (accordion model)

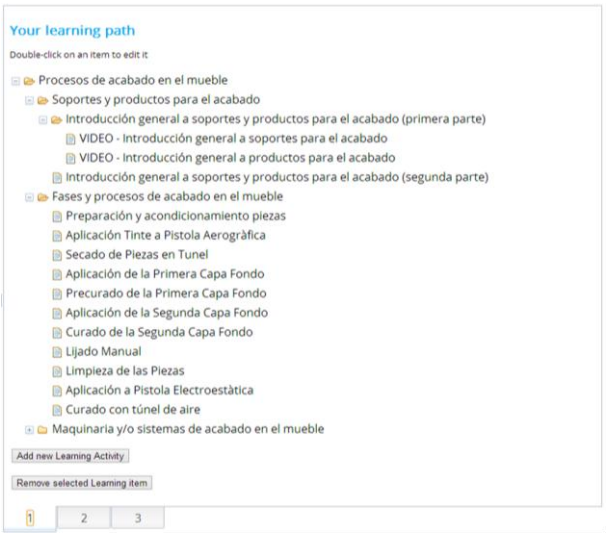
Your learning path
Double-click on an item to edit it
Procesos de acabado en el mueble
Soportes y productos para el acabado
Introducción general a soportes y productos para el acabado (primera parte)
VIDEO - Introducción general a soportes para el acabado
VIDEO - Introducción general a productos para el acabado
Introducción general a soportes y productos para el acabado (segunda parte)
Fases y procesos de acabado en el mueble
Preparación y acondicionamiento piezas
Aplicación Tinte a Pistola Aerogràfica
Secado de Piezas en Tunel
Aplicación de la Primera Capa Fondo
Precurado de la Primera Capa Fondo
Aplicación de la Segunda Capa Fondo
Curado de la Segunda Capa Fondo
Lijado Manual
Limpieza de las Piezas
Aplicación a Pistola Electroestàtica
Curado con túnel de aire
Maquinaria y/o sistemas de acabado en el mueble
Add new Learning Activity
Remove selected Learning item
1
2
3

Figure 29. Activity Widget (A-WDGT) prototype (tree model)

The tree model has been also equipped with some basic editing functionalities to add and remove items from the tree and to move items along the tree with a simple drag and drop functionality. Although direct editing is not listed among the required features, such functionalities could be exploited to allow the trainee to express a refinement of the mix composed and could support the editing of predefined mixes by trainer in the backend layer.

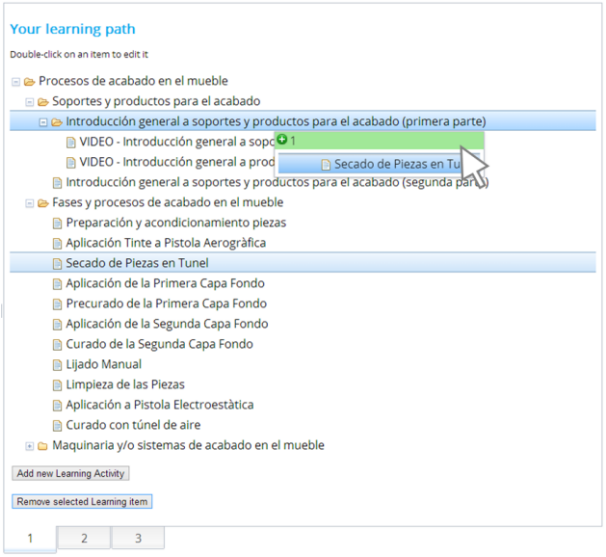
Your learning path
Double-click on an item to edit it
Procesos de acabado en el mueble
Soportes y productos para el acabado
Introducción general a soportes y productos para el acabado (primera parte)
VIDEO - Introducción general a sopc
VIDEO - Introducción general a prod
Secado de Piezas en Tu
Introducción general a soportes y productos para el acabado (segunda parte)
Fases y procesos de acabado en el mueble
Preparación y acondicionamiento piezas
Aplicación Tinte a Pistola Aerogràfica
Secado de Piezas en Tunel
Aplicación de la Primera Capa Fondo
Precurado de la Primera Capa Fondo
Aplicación de la Segunda Capa Fondo
Curado de la Segunda Capa Fondo
Lijado Manual
Limpieza de las Piezas
Aplicación a Pistola Electroestàtica
Curado con túnel de aire
Maquinaria y/o sistemas de acabado en el mueble
Add new Learning Activity
Remove selected Learning item
1
2
3

Figure 30. Activity Widget (A-WDGT) prototype (tree model with drag&drop editing functionalities)

The drag & drop functionality allows to reorder learning activities along the learning path and can be potentially exploited to allow the trainee to customize the proposed learning path, providing feedback to the system, whereas the Add/Remove functionalities are primarily meant to allow the manual creation and editing of learning paths for trainers.

4.3.3.7 Conclusion and Future Plans

Not applicable at the current stage of design/development.

4.4 Licensing

See "Licensing" sections for each specific component of the DW IOS platform.

4.5 Technical Manual

See "Technical Manual" sections for each specific component of the DW IOS platform.

4.6 User Manual

See "User Manual" sections for each specific component of the DW IOS platform.

4.7 Conclusion and Future Plans

After the delivery of the first version of the prototypes (M12) and their integration (M15) , the DW IOS platform components will undergo a validation phase that will provide information on how to improve / add functionalities, both from technical and ease-of-use point of view. Technical and functional documentation will be updated accordingly.

5. Virtual World IoC Platform

5.1 Overall data

Component Name	**VW IOC**
Reference WP – Task	WP2 – Task 2.3
Responsible	PIKS
Version	0.1
Source Control	https://www.tellme-ip.eu/svn-TELLME/TELLME
Contact Person	Maurizio Megliola (mailto:maurizio.megliola@piksel.com), Gianluigi Di Vito (gianluigi.divito@piksel.com)
Short Description	Service platform for Networked Media Channeling and Ingestion

5.2 Architecture and Functionalities

5.2.1 Introduction

The goal of this component is to provide a service platform for Networked Media ingestion and channeling. This includes media ingestion facilities for acquiring new audiovisual contents (e.g.: through mobile devices) to be properly transcoded and stored within the TELL-ME platform, making them available for further annotation through the media repository configuration interface provided in WP3. Moreover it includes multi-channel delivery capabilities by matching the consumption device characteristics (e.g.: screen size, bandwidth, etc.) with the available media contents (e.g.: low definition video, high definition video, etc.).

On the client side it provides the Media Player component, vertically designed and developed in order to support the further integration in a Widget/Gadget/HTML-based container constituting the TELL-ME entry point for the end-user.

Task T2.3 will build on existing knowledge emanating from the NoTube (distributed repository of Multimedia contents and ingestion of heterogeneous content soruces) and IKS EU projects (streaming/playback of annotated media contents).

The PIKSEL Video CMS platform is taken as the reference architecture for the VW IOC platform in TELL-ME, connecting to the media configuration tools provided in WP3.

5.2.2 Summary of Functionalities

Functionality Name	**Media Ingestion**
Description	Component providing multimedia ingestion capabilities. For the first prototype it will be IP-based and the files to be ingested will be shared on a remote file system to be further processed for transcoding and storage within the TELL-ME repository.
Inputs and source	The media file to be ingested or the media URI in case remote ingestion. Source(s): mobile devices or cameras.

Output and target	No output foreseen. Target(s): the Polymedia Media Asset Management platform + proprietary components for transcoding, the TELL-ME Repository Data Access Layer (WP3) for storage.
Interface details	• Remote file system access • RESTful service for ingesting a binary media files • RESTful service for ingesting a file located on a remote file system

Functionality Name	**Multi-Channel Streaming Engine**
Description	Component providing multichannel streaming capabilities for media contents. For the first prototype it will be based on an Apache Web Server + the H264 Streaming Module for Apache. Back-end services for requesting the video content basing on the considered context (desired video quality, workplace characteristics, etc.) will be developed supporting the integration of the player.
Inputs and source	1. ID Learning Content , ID User (linked to Workplace and Device) 2. ID Learning Content, Desired quality 3. ID Learning Content, ID Device, ID Workplace Source(s): the Media Player as well as any other front-end component requiring streaming media.
Output and target	The streamed media content using the H264/MPEG-4 AVC protocol. Target: widget-based UI
Interface details	A set of RESTful services implementing the input/output requirements described above.

Functionality Name	**Media Player**
Description	Vertical component providing multimedia playback functionalities. The first prototype will be based on a HTML5-based Web Player leveraging the *popcorn.js* framework from Mozilla. This will ensure the usage of commonly adopted Web standards as well as interoperability with popular browsers as well as mobile devices (running Android and iOS). This choice enables also the extension of the UI module with minimal effort by adapting or wrapping the produced code in order to support the integration with the platform UI container (under investigation by the Consortium at the time of reporting)
Inputs and source	The Learning Content ID referring to media content (to be played) Source(s): reasoning components (WP2), the user in case of explicit interaction with the interface presenting a list of media content related to his/her search results.
Output and target	To the user: playback of the chosen media content To the Streaming Engine: ID Learning Content, desired video quality / ID Workplace / ID User (according to the interfaces exposed by the engine – see above) Target(s): the user (trainee) for the video consumption, the Streaming Engine + back-end services for software integration
Interface details	HTML page + parameters (GET/POST) depending on the integration requirements of UI container

5.2.3 Architecture

Below the general architecture of the VW IoC platform is presented highlighting the modules involved in the TELL-ME platform, inlcuding connection with other WPs (mainly WP3, providing access to media contents as well as facilities and services for preparing them) and the components developed in the scope of this task, in particular:

- The Streaming Engine, part of the platform delivery facilities
- The Video Player, focusing on the Web/Mobile channels
- The Media Ingestion, related to images, videos and potentially network-based external sources (the different workplaces)

The complete technological stack is included in the diagram even though some functionalities are not foreseen in light of the current requirements, but can be further extended in the future by leveraging the PIKSEL Video CMS suite. Needless to say some services or facilities have to be carefully evaluated in light of the impact they may have during the integration of the final prototype (e.g.: DRM).

The connection with the TELL-ME repository is provided thanks to the Data Access Layer services (see WP3 for more details). Locally, the IoC platform will be able to manage Low Quality (LQ)/High Quality (HQ) contents as well as associated Metadata.

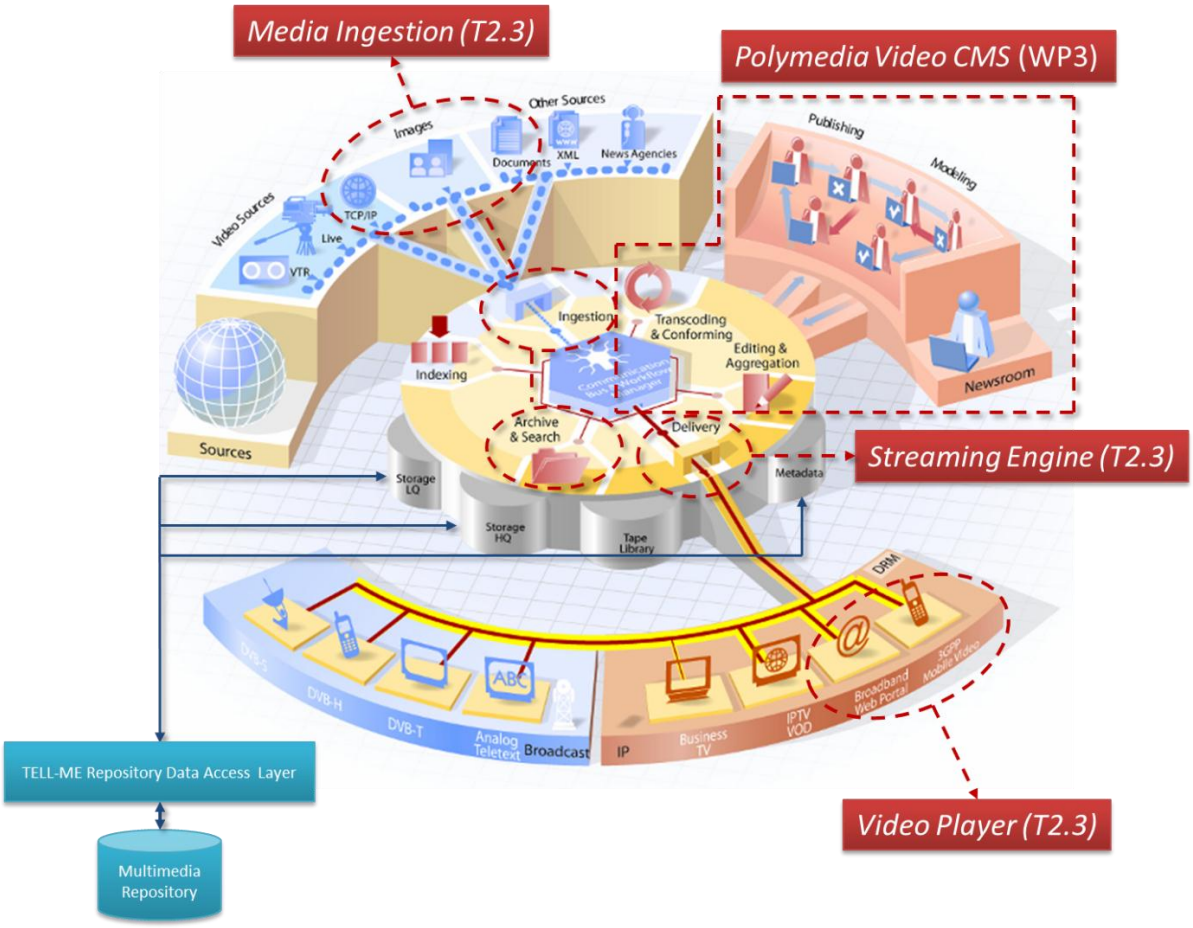

Figure 31 – VW IOC General Architecture

5.3 Technical Information

Nature	Media Ingestion
Programming Language	Java, proprietary scripting language
Development Tools	Eclipse
Additional libraries	PIKSEL Video CMS back-end
Application Server	Tomcat
Databases	Oracle DB for the Polymedia Platform, Data Access Layer for the TELL-ME platform

Nature	Video Player
Programming Language	HTML5, javascript
Development Tools	Eclipse/Any HTML Editor
Additional libraries	popcorn.js (http://popcornjs.org/)

Application Server	Dependant to the UI container (requirement: Web Server)
Databases	None – Relying on the Streaming Engine

Nature	Streaming Engine
Programming Language	Java
Development Tools	Eclipse
Additional libraries	json/jsonp-based, libraries supporting RESTful Services
Application Server	Apache Web Server + the H264 Streaming Module for Apache (currently evaluating alternatives such as: TVersity, Windows Media Server)
Databases	Data Access Layer for the TELL-ME platform

5.4 Licensing

- **PIKSEL Video CMS**: commercial product (proprietary)
- **Eclipse**: Eclipse Public License (EPL). It's an open source software license used by the Eclipse Foundation for its software. It replaces the Common Public License (CPL) and removes certain terms relating to litigations related to patents. The Eclipse Public License is designed to be a business-friendly free software license and features weaker copyleft provisions than contemporary licenses such as the GNU General Public License (GPL). The receiver of EPL-licensed programs can use, modify, copy and distribute the work and modified versions, in some cases being obligated to release their own changes. The EPL is approved by the Open Source Initiative (OSI) and is listed as a free software license by the Free Software Foundation (FSF).
- **Tomcat**: Apache License, Version 2.0. The Apache License is a free software license written by the Apache Software Foundation (ASF). The Apache License requires preservation of the copyright notice and disclaimer. Like any free software license, the Apache License allows the user of the software the freedom to use the software for any purpose, to distribute it, to modify it, and to distribute modified versions of the software, under the terms of the license, without concern for royalties.
- **Apache Web Server** + the *H264 Streaming Module* for *Apache*: Apache License, Version 2.0 (see above). The H264 Streaming Module for Apache, is an additional component of the Apache Web Server; in light of the ASF, all software produced by the ASF or any of its projects or subjects is licensed according to the terms of the Apache License.

5.5 Technical Manual

Instructions about the deployment of the developed software components for the first prototype implementation are reported in the following tables.

Functionality Name	**Media Ingestion**
Pre-condition	Tomcat 6.x or 7.x, Piksel Video CMS
How to install	The Web services that allow the communication with the Piksel Media Asset Management platform are provided as a war file; the file must be installed using Apache tomcat 6.x or 7.x.
Deploy	The deploy can be performed using the Tomcat Web Application Manager, or manually by copying the war file in the folder "CATALINA_HOME / webapps /" and restarting the server.

Functionality Name	**Multi-Channel Streaming Engine**
Pre-condition	Tomcat 6.x or 7.x, Piksel Video CMS
How to install	The Web services that allow the communication with the Piksel Streaming Engine are provided as a war file; the file must be installed using Apache tomcat 6.x or 7.x.
Deploy	The deploy can be performed using the Tomcat Web Application Manager, or manually by copying the war file in the folder "CATALINA_HOME / webapps /" and restarting the server.

Functionality Name	**Media Player**
Pre-condition	Web Browser compatible with HTML 5 video TAG
How to install	The media player is a web component, provided in a zip file. In order to use it, it must be unzipped under a web server. The video player expects, as input parameter (videoed), the id of the media content to show. Once recovered the id, by a webservice calling, it retrieves the video url and all the metadata associated with it. For instance: "tellmevideo.html?videoid=152".
Deploy	-

5.6 User Manual

Not applicable at the time of reporting.

5.7 Conclusion and Future Plans

The above sections provide specifications of the VW IoC platform for the first prototype implementation due M12. A minimum set of the three core functionalities (ingestion, streaming and playback) has been provided as the basis for further refinements/extensions to them during the rest of the project in light of any additional or updated requirements coming from the Consortium partners and user trials. The platform will be integrated by M15 with the CORE part of the TELLLME system.

6. Conclusions and Next Steps

In this document the architectural components specifications of WP2 DW-RW-VW platforms have been provided in order to accompany the technical prototypes (The deliverable is in fact correctly described as "P" in the TELL ME DOW). RW provides an IOT technology stack able to automatically detect learning needs. DW manages the service mashup helping the user in the control of data flow. VW platform provides a system to stream and ingest multimedia contents. For each of the three major components the following information has been reported:

- *Overall Data* in order to provide pointers to the information needed
- *Introduction* to the tool about its scope and inputs – outputs from other tools
- *Summary of Functionalities* describing major functionalities provided by the application
- *Architecture* of the tool
- *Technical Information* reporting technologies used during the implementation
- *Licensing of the tool* and of third party software like libraries
- *Technical Manual* about how to technically install and manage the prototype
- *User Manual* only for tools having a GUI to be used by a human
- *Conclusion and Future Plans*

In attach to the deliverable the specification of the task 2.4 coming from WD2.1 and the overall TELL ME architecture extracted from WD6.1 are provided to complement the set of information at the disposal of the reader.

The three components have been implemented by M12 and will be integrated in the whole architecture by M15. The integration will take place partially in WP2 (intra-WP2 components) and partially in WP6 (inter WPs components). The integrated system will be the major result of the M15 project milestone.

7. Annex A: Middleware integration layer (WD2.1 extract)

7.1 Overall data

Component Name	**TELL ME CORE**
Reference WP – Task	WP2 – Task 2.4
Responsible	TXT
Version	M15 will report version 1.0
Source Control	Available in M15
Contact Person	Michele Sesana (michele.sesana@txtxgroup.com)
Short Description	This component will integrate the three platforms (RW, DW, VW) and will assure privacy, security and interoperability among the different components of the TELL ME architecture.

7.2 Architecture and Functionalities

7.2.1 Introduction

This component will integrate the three platforms (RW, DW, VW) and will assure privacy, security and interoperability among the different components of the TELL ME architecture.

7.2.2 Summary of Functionalities

The WP2 will be integrated by a middleware layer that should be able to provide both reasoning functionalities to the whole system acting as TELL ME CORE and features like: privacy, security and interoperability.

Major functionalities of the system are:

- Mediation among services including protocol mediation, payload mediation, etc;
- Privacy in data access: the identity of the user is checked in order to allow the access just on data belonging from his/her domain and for which had the authorisation;
- Security: the service layer is provided by a SaaS approach; the security of data is a key aspect in order to be able to run it in working environment with real data. For example calls to system services are allowed just for specific servers filtered by IP address, a limit of calls can be set;
- Monitoring of the system performance is important in order to understand the response time of the system and the workload of servers to be improvbed in the second prototype .
- Business execution process: a BPEL execution environment is provided for the execution of CORE processes like the parallel research of learning contents in different data sources, the execution of business rules for the automatic creation of mixes, etc.

7.2.3 Architecture

The item is composed by several subcomponents collaborating together in order to assure the complete control of the data flow.

- Enterprise Service Bus (ESB): every call to a service in the TELL ME system is intercepted by a virtual endpoint in an enterprise service bus providing mediation and implementing privacy and security
- The ESB is integrated with an Identity Service that will provide security and privacy being able to discriminate users
- A BPEL engine takes in charge of the platform services orchestration especially in relation with the answer to learning needs
- The Business Rule Engine will provide the reasoning to the BPEL processes

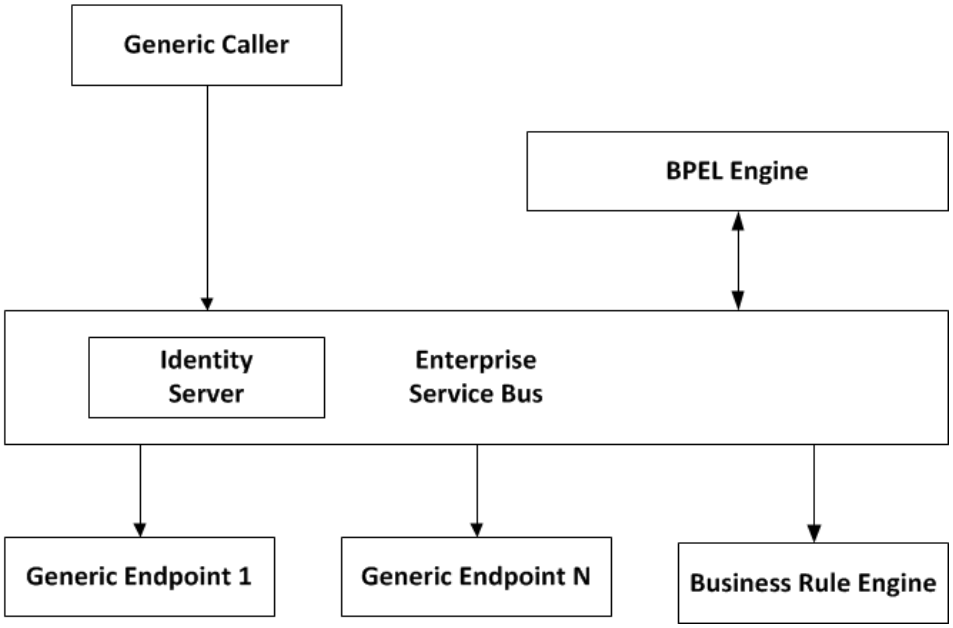

Figure 32: Internal Architecture of T2.4

7.3 Conclusion and Future Plans

The prototype is under development and will be extensively reported in D2.3.

Sections related to the prototype will be fulfilled in that document (Technical Information, Licensing, Technical Manual, User Manual, Conclusion and Future Plans).

8. Annex B: TELL ME Architectural Design (WD6.1 extract)

This chapter provides an extract from Working Document WD6.1 providing the specifications of the first version of the TELL ME architecture. This summary is composed by

- Introduction in chapter 8.1
- In chapter 8.2 the actors of the system and the associated use case diagram
- chapter 8.3 the overview of the architecture

8.1 Introduction

D6.1 is the Prototype of the first integrated system (WP2+WP3+WP4) implemented and will have as accompanying deliverable as update of working document WD6.1 from which this text is extracted reflecting possible future changes and some information about the prototype implementation such as: availability of the source code, contact persons, technical manual, licenses, etc.

8.2 System Actors and Use-case Diagram

Major TELL ME **actors** are:

- **Trainer:** the person that knows the job and set the system (could be as well the boss of the worker)
- **Trainee** (Blue Collar Worker)**:** the blue collar worker using the TELL ME system to accomplish his/hers job
- **Smart Environment**: smart part of the workplace environment (AmI – IOT – etc.) providing contextual – real time information about the user actions and environmental conditions
- **Community**: composed by trainers and trainees coming from one or more workplaces

In WP8 the actors will be refined in order to associate existing company roles to the above mentioned roles and add new actors. For example the quality managers are now associated to trainers but a specific role can be created during customisation.

Use-cases diagram

- The trainer should be able to **configure** the system for example defining rules for the automatic detection of learning needs, model the re-usage of existing learning contents, etc.
- **Learning Needs (eMeMO *enquiry*)** to be challenged and solved are **reported** by different actors in both push and pull modalities:
    - by the trainee typically using a push interface;
    - by the trainer looking at the worker behaviour and/or to information received from the system ;
    - triggered automatically by the smart environment or scheduled by the trainer;

- based on the configuration and the learning need, the TELL ME system **provides** to the worker the **learning path (eMeMO *Mix*)** and the learning contents (LCOs) to learn how to accomplish the job
- the trainee after receiving the package and the contents should be able to **consume** them to then accomplish the job
- during the learning by doing experience (eMeMO *experience*) the trainee activities and achievements are **observed** by the system; of course accordingly to national privacy legislations
- trainee and the community can provide **feedbacks** on LP and LCOs provided by the system and about how they are provided (for example: are they useful? Provided in the right context? Provided in the right technical way?)

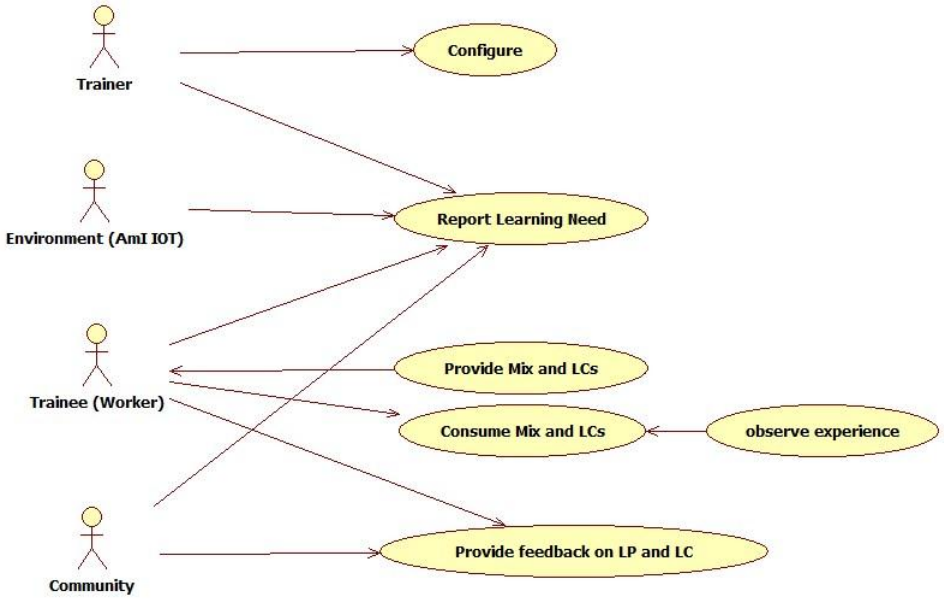

Figure 33: TELL ME use-cases diagram

8.3 Architecture Overview

The TELL ME architecture is designed as a modular system in order to be then instantiated in (possibly) any kind of workplace in various forms. Of course, in the scope of the project the platform will be instantiated, experimented and evaluated within the three project workplaces:

- helicopter maintenance (AW);
- furniture manufacturing and assembly (AIDIMA);
- textile inspection (PROFI).

Here below the conceptual high level architecture coming from the DOW is reported to very rapidly recap the starting point for WP6 activities; afterwards the first version of the architecture is reported.

8.3.1 Logical Architecture (DOW)

The TELL ME architecture is based on three different levels further detailed and implemented by WP2, WP3 and WP4. In Figure 34 the logical view of the architecture described in the TELL ME DOW [DOW] is reported. The lower layer (in blue) takes care of the management of learning contents; in orange the WP2 is visible providing the service infrastructure with the four platforms addressing the real, digital and virtual worlds coupled with the secured interoperability infrastructure. The upper layer (in green) encompasses the workplace including ergonomics, mobile and wearable devices and multimodal interfaces. This is the level to which the worker interacts.

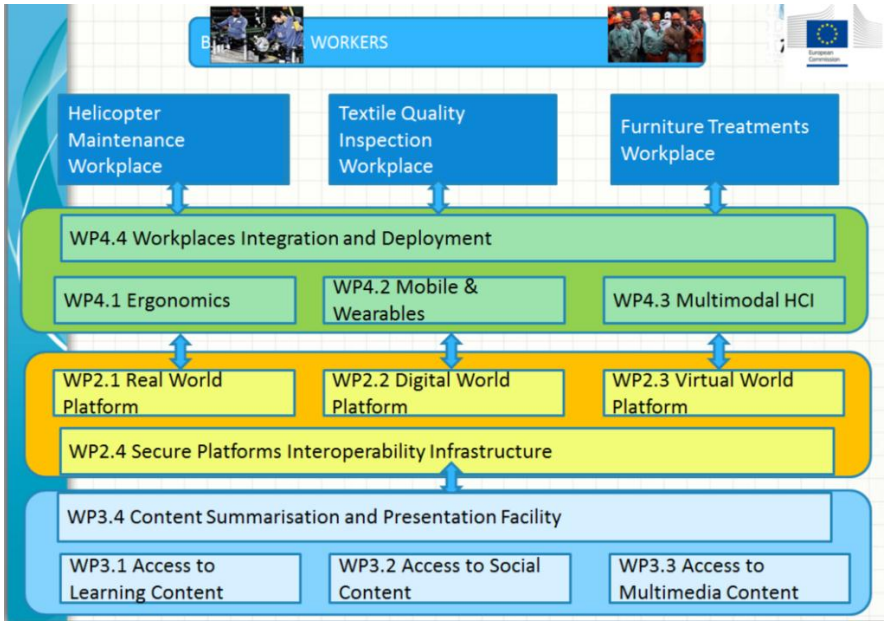

Figure 34: TELL ME Architecture - DOW version

8.3.2 First Architecture

In Figure 2 the first version of the TELL ME technical architecture is reported.

As introduced before conceptually the architecture is conceptually based on three layers:

- user interaction layer including the workplace and the user/ambient interaction with TELL ME
- middleware providing access to all the TELL ME services and managing data flow and implementing the CORE of the system
- data layer composed by a set of repositories and corresponding APIs to access and manage them

Here below Figure 35 shows the first version of the TELL ME architecture and subsequently the different layer are duly described.

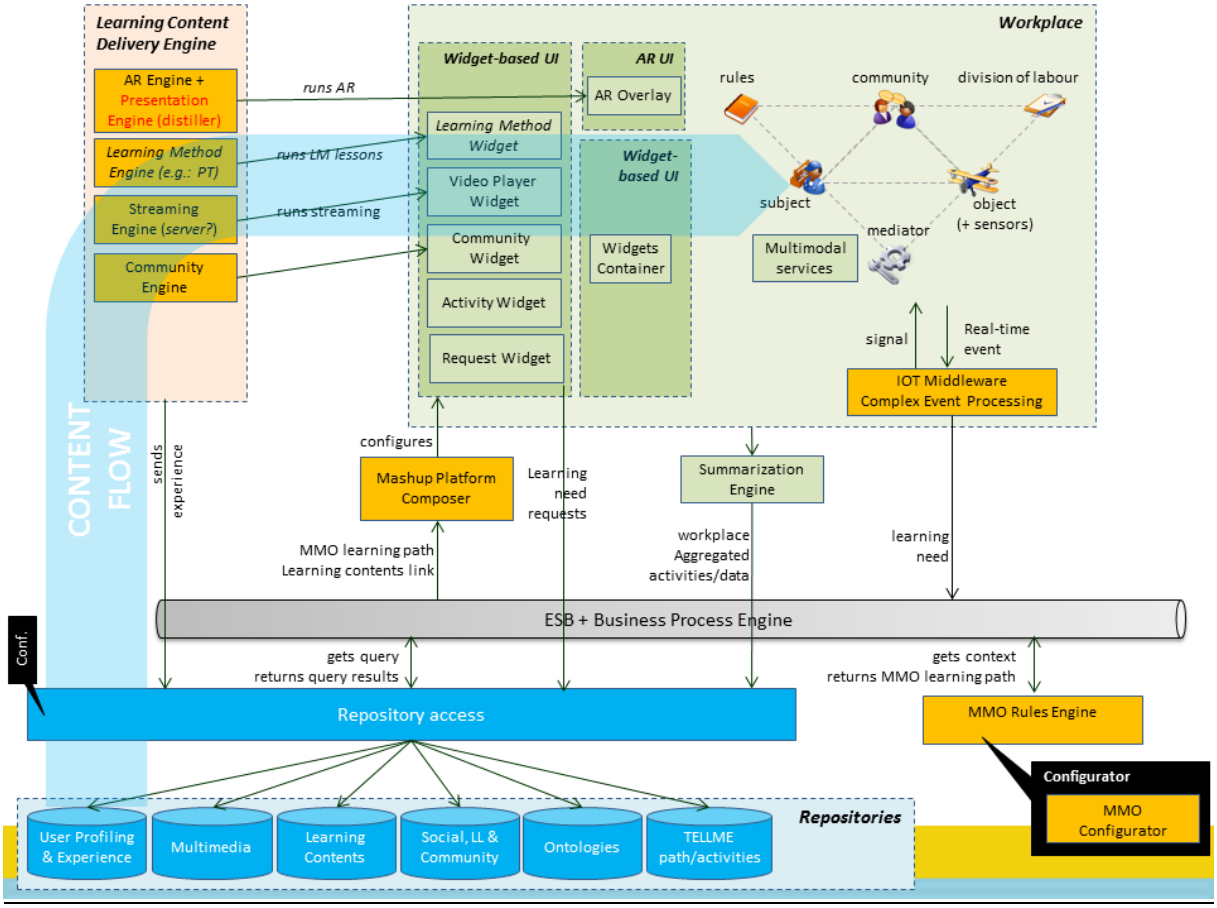

Figure 35: First version of the TELL ME architecture

User Interaction Layer at the top the smart workplace is visible and described following the activity theory (please refers to TELL ME WD4.1 and D4.1 for more information).

The user interacts with the real workplace environment that can be equipped with sensors and actuators, a digital environment for accessing the TELL ME system functionalities and a virtual/augmented immersive world providing an interaction layer on top of the physical environment.

The TELL ME digital system functionalities are offered to the user by a number of HTML5 components that are deployed in several containers depending of the kind of workplace addressed. For the first prototype three containers are supported: augmented reality, an HTML5 environment created ad-hoc for the project and a widget based container. The interaction with the system functionalities can be done by intuitive and dynamic Graphic User Interface (GUI) enhanced by multi-modal interfaces (speech, gesture recognition, text-to-speech, etc.). The containers will be displayed by a various set of mobile and static devices.

A set of engines complements the User Interaction Layer providing the GUI with the ability to runs different kind of learning contents; for example, the streaming server supports the video consumption, the Precision Teaching engine provides the possibility to run Precision Teaching lessons, etc.

Real world interfaces: The user interaction layer is not based only on digital interfaces; the real world can be equipped by a set of ambient intelligence/IOT devices (sensors, actuators, etc) that can records user actions to be investigated in the middleware layer and, vice-versa informs the user about something happening.

Middleware Between the presentation and the storage is the middleware layer that provides all the services in order to manage contents, runs the WP1 meta-methodology, monitor and store user sessions, etc. The level provides as well interoperability, security and privacy in services access being composed by off the shelf middleware such as a BPEL engine, an Enterprise Service Bus (ESB), a rule engine and an identity server.

Data Layer The lower layer of the architecture is composed by a set of interconnected repositories providing the information necessary to the composition of the customized and context-based learning material to be submitted to the user.

- *User profile and experience* storing the user static profile and the dynamic information about experiences done by the user and logged by the system
- *Multimedia repository* containing multimedia tagged learning contents like videos, pictures, audio and video with audio track.
- *Learning Contents* providing tagged non-multimedia learning contents like documents (different formats), SCORM packages, etc.
- Social data
- *Ontologies* (cross-domains or domain-specific ontologies)
- *TELLME path/activities* repository storing specific TELL ME data such as mix, system configuration, etc.

GUI and APIs for accessing repositories and algorithms are encapsulated in the “repository access” component. Co-creation of the learning material (for example video ingestion and tagging) is managed as well by these configuration interfaces.

8.3.3 Link to WPs implementing the architecture

The link between the technical work-packages and the architecture is not 1-to-1 but can be seen as:

WP4 implements the workplace (T4.1) and the human.computer interaction (augmented reality – multimodaal interactions services and devices - T4.2, T4.3)

WP2 implements the middleware layer (T2.4) , the real world (IOT – T2.1), the digital world (GUIs, request of learning need and provision of mixes - T2.2) and the multimedia environment (T2.3)

WP3 supports the repository layer in which all the necessary internal data are stored, external sources are references and data access and management is provided. Interfaces are at service level to the middleware layer and, by GUIs, to the trainer.

8.3.4 Learning Contents Flow

In the following picture a different view of the architecture is provided to better show the flow of learning material.

- During activities in the workplace in push or pull modalities some learning needs are generated (request blue arrow)
- In the bottom right corner the **Decision Making** area of the architecture is visible; it selects the best learning contents to be supplied to the user;
- The selection is based on the information contained in the **Learning Contents Storage**. It includes user profiles, context, Learning Contents, domain ontologies, etc. This part of the architecture is accessed by the decision making part to have the whole set of information to make a decision and by the learning provision to retrieve the contents.
- In the top-left corner the **Learning Contents Provision** to the worker in the workplace is visible. It is composed by the presentation layer, the back end layer composed by the engines of the different UIs and the orchestrator “mashup platform composer” that can be seen as well as the Activity and Request applications engine.
- In the top right corner in green is visible the workplace and **Learning Contents Consumption** where the user works and consumes the material provided.

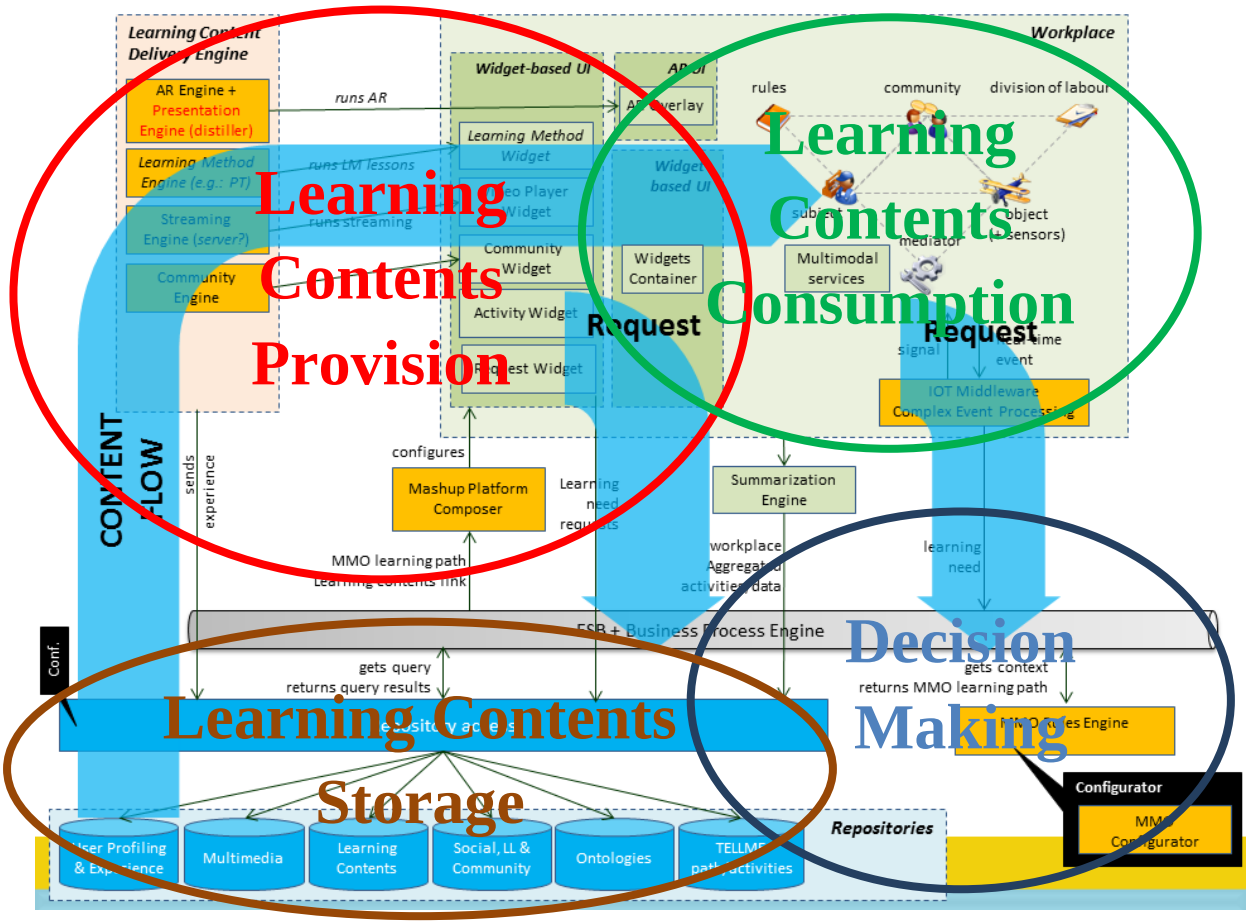

Figure 36: TELL ME architecture – Learning Contents Flow